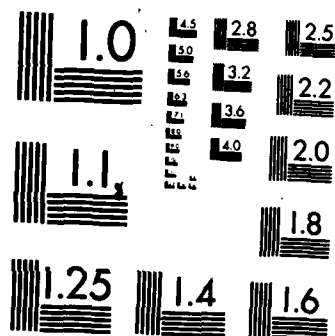


AISIM (AUTOMATED INTERACTIVE SIMULATION MODEL) TRAINING 1/2
EXAMPLES MANUAL(U) HUGHES AIRCRAFT CO FULLERTON CA
GROUND SYSTEMS GROUP M DESHLER ET AL. 26 FEB 82

ESD-TR-83-254 F19628-79-C-0153

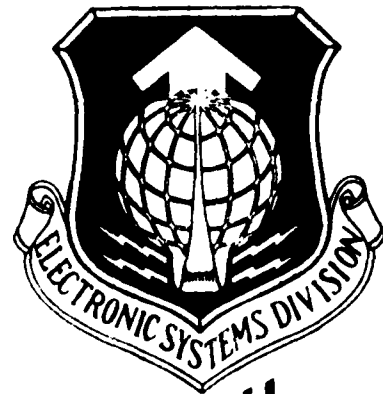
F/G 9/2

NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

12



AD A137719

AISIM TRAINING EXAMPLES MANUAL

M. Deshler
S. Kneeburg

Hughes Aircraft Company
Ground Systems Group
Box 3310
Fullerton, CA 92634

Approved for public release; Distribution unlimited

26 February 1982

DTIC
ELECTE
FEB 10 1984
S B D

Prepared for

ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
DEPUTY FOR ACQUISITION LOGISTICS
AND TECHNICAL OPERATIONS
HANSCOM AIR FORCE BASE, MASSACHUSETTS 01731

DTIC FILE COPY

84 02 10 008

LEGAL NOTICE

When U.S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.


OTHER NOTICES

Do not return this copy. Retain or destroy.

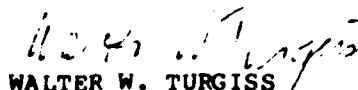
Review and Approval

This technical report has been reviewed and is approved for publication.


WILLIAM J. LETENDRE
Program Manager, Computer Resource
Management Technology (PE 64740F)


ARTHUR G. DECELLES, Capt, USAF
Project Officer, Requirements
Analysis

FOR THE COMMANDER


WALTER W. TURGISS
Director, Engineering and Test
Deputy for Acquisition Logistics
and Technical Operations

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM									
1. REPORT NUMBER ESD-TR-83-254	2. GOVT ACCESSION NO. AD A-011-1	3. RECIPIENT'S CATALOG NUMBER									
4. TITLE (and Subtitle) AISIM Training Examples Manual		5. TYPE OF REPORT & PERIOD COVERED CDRL No. 120									
		6. PERFORMING ORG. REPORT NUMBER									
7. AUTHOR(s) M. Deshler S. Kneeburg		8. CONTRACT OR GRANT NUMBER(s) F19628-79-C-0153									
9. PERFORMING ORGANIZATION NAME AND ADDRESS Hughes Aircraft Company Ground Systems Group Box 3310, Fullerton, CA 92634		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS									
11. CONTROLLING OFFICE NAME AND ADDRESS Hq Electronic Systems Division (ALRE) Hanscom AFB Massachusetts 01731		12. REPORT DATE 26 February 1982									
		13. NUMBER OF PAGES 136									
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) Unclassified									
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE									
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.											
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)											
18. SUPPLEMENTARY NOTES											
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) <table border="0"> <tr> <td>AISIM</td> <td>Methodology</td> <td>Construct</td> </tr> <tr> <td>Training</td> <td>Design</td> <td>Analysis</td> </tr> <tr> <td>Examples</td> <td>Plan</td> <td></td> </tr> </table>			AISIM	Methodology	Construct	Training	Design	Analysis	Examples	Plan	
AISIM	Methodology	Construct									
Training	Design	Analysis									
Examples	Plan										
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) <p>This document is the training examples manual for the Hughes developed Automated Interactive Simulation Model (AISIM). This manual describes how to translate a problem statement into a simulation model, utilizing the design and construction of example problems.</p>											

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CONTENTS

1.	Introduction.....	1
1.1	Purpose.....	1
1.2	Scope.....	1
1.3	Organization.....	1
1.4	Documentation Conventions.....	2
1.5	Applicable Documents.....	2
2.	AISIM Methodology.....	3
2.1	Background.....	3
2.2	Inputs to the AISIM Simulation Process.....	5
2.2.1	Mission Concept and Requirements.....	5
2.2.2	Problem Perspective.....	6
2.2.3	System Description.....	6
2.3	Preliminary Analysis.....	6
2.3.1	Justifying AISIM Simulation.....	6
2.3.2	Define the Problem and Objectives.....	7
2.3.2.1	Defining the Problem.....	8
2.3.2.2	Defining the Simulation Objectives.....	8
2.4	Design, Plan and Construct the Model.....	8
2.5	Exercise the Model.....	9
2.5.1	Single Thread Scenarios.....	10
2.5.2	Multiple Thread Scenarios.....	10
2.5.3	Analysis Scenarios.....	11
2.6	Analyze Results.....	11
2.6.1	Identify Measured Performance Statistics.....	11
2.6.2	Validate the Model.....	11
3.	Example 1 - Communication Network.....	13
3.1	Input.....	13
3.1.1	Mission Concept.....	13
3.1.2	Problem Perspective.....	13
3.1.3	System Description.....	14
3.1.3.1	Proposed Network Design.....	14
3.1.3.1.1	Message Flow.....	15
3.1.3.1.2	Routing.....	15
3.1.3.1.3	Channel Capacities.....	17
3.1.3.1.4	Nodal Processing.....	17
3.1.3.2	Message Traffic.....	19
3.2	Preliminary Analysis.....	23
3.2.1	Justifying AISIM Simulation.....	23
3.2.2	Problem Definition.....	23
3.2.3	Definition of Objective.....	25
3.3	Model Build.....	25
3.3.1	Design, Plan and Construction of the Model.....	25
3.3.1.1	Model Design.....	25
3.3.1.2	Model Implementation Plan.....	26

3.3.1.3	Model Construction.....	27
3.3.1.3.1	Message Routing.....	27
3.3.1.3.2	Process: REQ-I/O.....	29
3.3.1.3.3	Process: ESR-CALL.....	33
3.3.1.3.4	Process: ROUTER.....	34
3.3.1.3.5	Process: CONTROL.....	37
3.3.1.3.6	Process: CHLIO.....	41
3.3.1.3.7	Process: IHANDLER.....	43
3.3.1.4	Subset of System.....	46
3.3.1.4.1	Process: HCOPICHQ.....	43
3.3.1.4.2	Process: CHQHD.....	49
3.3.1.4.3	Single Thread Trace.....	51
3.3.1.5	Complete Network Architec- ture.....	52
3.3.1.6	Node Process Definitions.....	53
3.3.1.6.1	Processes DATAB01 through DATAB07.....	53
3.3.1.6.2	Process: BECHO.....	54
3.3.1.6.3	Process: B-ORIGIN.....	55
3.3.1.7	Processes: DATABCHQ, DATABHQ1 and DATABHQ2.....	56
3.3.1.7.1	Processes: HQ1DGEN and HQ2DGEN.....	59
3.3.1.7.2	Processes: HQ1GGEN and HQ2GGEN.....	60
3.3.1.8	Processes: HQ1HGEN and HQ2HGEN.....	61
3.3.1.9	Process: CHQGD.....	62
3.3.2	Load: LOADS01.....	63
3.4	Analysis of Results.....	64
3.4.1	Performance Measures.....	64
3.4.1.1	Communications Queue Time.....	64
3.4.1.2	Communication Channel Queue Length.....	65
3.4.1.3	Processor Queue Time.....	65
3.4.1.4	Processor Queue Length.....	65
3.4.1.5	Communications Channel Utiliza- tion.....	65
3.4.1.6	Processor Utilization.....	65
3.4.1.7	Other Performance Measures.....	65
3.4.2	Validating the Model.....	66
3.5	Conclusions.....	67
4.	Example 2 - Loop Communication Network.....	68
4.1	Input.....	68
4.1.1	Mission Concept.....	68
4.1.2	Problem Perspective.....	68
4.1.3	System Description.....	68
4.2	Preliminary Analysis.....	69
4.2.1	Problem Definition.....	70
4.2.2	Definition of Objective.....	70
4.3	Model Build.....	70

4.3.1	Design, Plan and Construction of the Model.....	70
4.3.1.1	Model Design.....	70
4.3.1.2	Model Implementation Plan.....	71
4.3.2	Define Pierce Loop Architecture.....	72
4.3.3	Interfacing Message Routing Submode to Loop Model.....	73
4.3.4	Process: GEN-MSG.....	73
4.3.5	Verifying the Message Routing Submodel and Loop Model.....	75
4.3.5.1	Single Thread Scenario and Load.....	75
4.3.5.2	Single Thread Trace.....	76
4.3.5.3	Multiple Thread Scenario and Loads.....	78
4.3.6	Full Loading Scenario.....	78
4.3.7	Extending the Model to Include Slots and Buffers.....	79
4.3.7.1	Entity Attributes.....	81
4.3.7.2	Process: STARTUP.....	83
4.3.7.3	Process: FORWARD.....	83
4.3.7.4	Process: GEN-MSG.....	85
4.3.7.5	Process REC-SLOT.....	87
4.3.7.6	Scenario and Loads.....	91
	4.3.7.6.1 Load: INITLOAD.....	91
	4.3.7.6.2 Load: LOADBl.....	91
4.4	Analysis of Results.....	91
4.4.1	Performance Measures.....	91
4.4.1.1	Queueing Time.....	92
4.4.1.2	Transmission Time.....	92
4.4.2	Validating the Model.....	92
4.5	Conclusions.....	92
5.	Bus Communication System Example.....	94
5.1	Input.....	94
5.1.1	Mission Concept.....	94
5.1.2	Problem Perspective.....	94
5.1.3	System Description.....	95
5.2	Preliminary Analysis.....	93
5.2.1	Justifying AISIM Simulation.....	93
5.2.2	Problem Definition.....	93
5.2.3	Definition of Objective.....	100
5.3	Model Build.....	100
5.3.1	Design, Plan and Construction of the Model.....	100
5.3.1.1	Model Design.....	100
5.3.1.2	Model Implementation Plan.....	101
5.3.1.3	Process: BIU.....	102
	5.3.1.3.1 Outgoing Data Request Message Handling.....	102
	5.3.1.3.2 Incoming Data Request Message Handling.....	103

5.3.1.3.3	Bus Grant Message handling.....	106
5.3.1.3.4	Data Message handling.....	107
5.3.1.4	Process: CONTRBUS.....	110
5.3.1.5	Process: NCEPROC.....	112
5.3.1.6	Process: DATA BUS.....	116
5.3.1.7	Process: SENDACK.....	117
5.3.1.8	Process: SENDREL.....	119
5.3.1.9	Process: HOST.....	120
5.3.1.10	Process: DRHOST.....	123
5.3.2	Load Drivers: Processes TOHOST1 through TOHOST6.....	125
5.4	Analysis of Results.....	126
5.4.1	Performance Measure.....	126
5.4.2	Validation.....	126

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



FIGURES

FIGURE		PAGE
1	Simulation Context.....	3
2	AISIM Simulation Process.....	4
3	Example 1 Suggested Topology.....	16
4	Example 1 Routing Tables.....	18
5	Example 1 Channel Capacities.....	19
6	Example 1 Processor Variables.....	19
7	Message Traffic Characteristics.....	21
8	Message Traffic Matrices.....	22
9	Example 1 Model Structure.....	26
10	Message Routing Structures.....	28
11	Message Routing Process Sequence.....	29
12	Example 1 Subset Architecture.....	47
13	Example 1 Single Thread Test Load and Scenario.....	48
14	Process HCOFYCHQ.....	49
15	Process CHQHD.....	50
16	Example 1 Single Thread Trace.....	51
17	The Complete Network Architecture.....	52
18	Example 1 Processes.....	53
19	Process DATABB01.....	54
20	Process BECHO.....	55
21	Process B-ORIGIN.....	56
22	Process DATABCHQ.....	57
23	Process DATABHQL.....	58
24	Process DATAHQ.....	59
25	Process HQLDGEN.....	60
26	Process HQLGGEN.....	61
27	Process HQLHGEN.....	62
28	Process CHQGD.....	63
29	Example 1 Load Originating at Node S01.....	64
30	Example 1 Node Utilization Results.....	66
31	Example 1 Channel Utilization Results.....	67
32	Pierce Loop Diagram.....	69
33	Example 2 Loop Architecture.....	72
34	Example 2 Legal Path Table.....	73
35	Example 2 NODE-TBL.....	74
36	Process GEN-MSG.....	75
37	Example 2 Single Thread Scenario and Load.....	76
38	Example 2 Single Thread Trace.....	77
39	Multiple Thread Test Case.....	78
40	Example 2 Full Loading Scenario and Entity Definitions.....	79
41	Example 2 Structure with Slot and Buffers.....	81
42	Example 2 process STARTUP.....	83
43	Example 2 Process FORWARD.....	85
44	Example 2 Process GEN-MSG.....	87
45	Example 2 Process REC-SLOT.....	88
46	Example 2 Load INITLOAD.....	91
47	Example 2 Load LOADB1.....	91
48	Bus Communication Protocol.....	97
49	Distribution of Message Destination by Source and t	99
50	Example 3 High Level Structure.....	101

51	Example 3 Bus Interface Unit Logic for Outgoing Data Request Message.....	103
52	Example 3 Bus Interface Unit Logic for Incoming Data Request Message.....	105
53	Example 3 Bus Interface Logic for Bus Grant Messages.....	107
54	Example 3 Bus Interface Logic for Data Message Handling.....	110
55	Example 3 Control Bus Message Handling Logic.....	112
56	Example 3 Network Control Logic.....	113
57	Example 3 Data Bus Message Handling Logic.....	117
58	Example 3 Acknowledge Message Generation.....	119
59	Example 3 Release Data Bus.....	120
60	Example 3 HOST Data Message Handling.....	121
61	Example 3 Data Request Message Generation.....	124
62	Example 3 Load Process TOHOST1.....	126

1. Introduction

1.1 Purpose

The Automated Interactive Simulation Model (AISIM) System provides the user with the ability to do high level simulation of complex operational and distributed data processing systems. The purpose of this manual is to provide the AISIM user with a set of examples which demonstrate the use of AISIM on problems which are typical of its application area. An AISIM methodology is presented and applied to each problem. Each example in this document is presented in such a way as to provide a standard for documenting an analysis effort using AISIM.

The emphasis of this manual is on how to translate a problem statement into a simulation model. For this reason, the focus of this manual is on model design and construction. Little attention is paid to the results of the simulation exercises. This approach is somewhat different from the one the user will take when developing AISIM models for analysis of systems because then the results will be of significant importance.

1.2 Scope

This manual describes the use of the AISIM applied to specific problems. The problems in this document have been selected based on the anticipated use of AISIM for the analysis of command, control and communication systems in the conceptual phase of development. The problems are presented in context to an emerging methodology based on AISIM simulation. Although the problems are specific to embedded computer communication systems, the methodology has a wider scope. AISIM can be used to analyze the dynamics of many types of systems. The techniques and strategies discussed in this document would be beneficial to anyone interested in discrete event simulation analysis.

The reader is expected to be a system's analyst who has experience analyzing conceptual designs of a multi-processing, computer based system. This document does not contain information necessary to operate AISIM nor does it address the hardware and software environments for AISIM. It is anticipated that this document will be read by a user after reading the AISIM Training Manual.

1.3 Organization

This manual is organized to be a teaching document for the AISIM user. Chapter 1 introduces this document, detailing the organization, the document conventions and applicable documents. Chapter 2 is a discussion of the AISIM methodology. Chapter 3 contains a detailed example of the use of AISIM to model a communications network. Chapter 4 contains a description of AISIM applied to another communication network using a loop

communication protocol. Chapter 5 contains an example of a communication system using a bus.

1.4 Documentation Conventions

The references in this document to specific words which are technical terms with meanings specific to AISIM that differ from generally accepted definitions will appear with an initial capital. This applies specifically to AISIM entities.

EXAMPLE: Process - occurrences of this word refer to the AISIM entity.

1.5 Applicable Documents

The following documents provide supporting information on the use of AISIM:

AISIM Training Manual

AISIM User's Manual

AISIM Product Specification

The following document provides supporting information on the examples described in this document.

AISIM Evaluation - Preliminary Report
Mitre Working Paper 23671

2. AISIM Methodology

Experience in using discrete event simulation for conceptual analysis has resulted in a procedure for simulating systems. This procedure is called "simulation analysis" and has been documented in various forms in simulation literature. Since AISIM is a discrete event simulation tool, all of the steps in the simulation analysis process apply to performing AISIM analysis. This discussion on AISIM methodology calls out the steps in the AISIM simulation process and points out how each of the steps is specifically performed. To put this into context, the background leading to the development of AISIM is described.

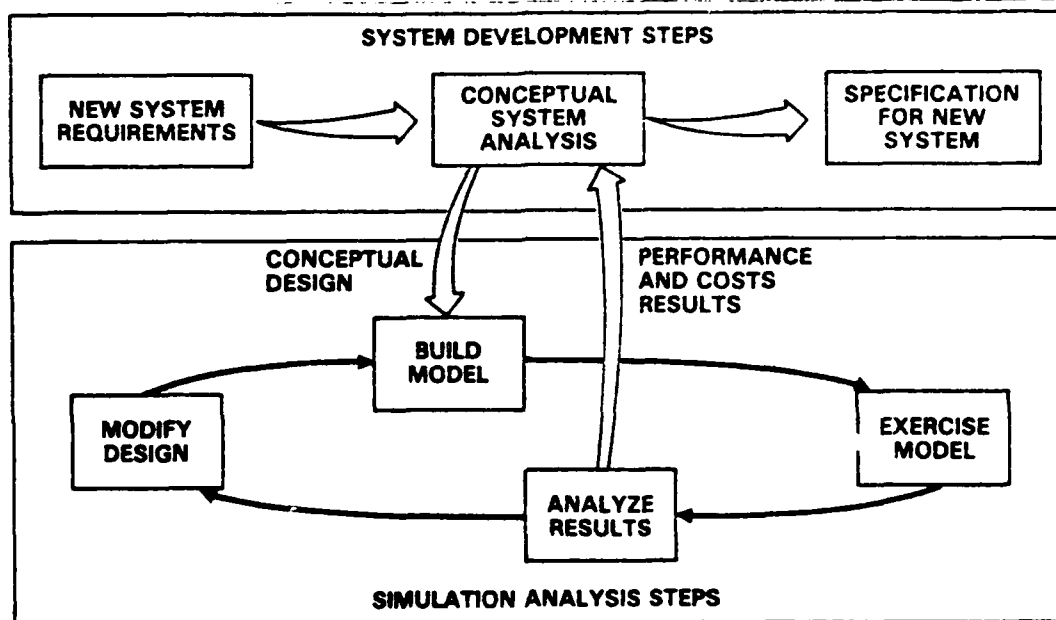


Figure 1. Simulation Context

2.1 Background

Simulation is a powerful analysis technique used to support conceptual analysis in order to specify requirements for a system. The goal is to engineer a system which works, that is, a system which will perform its mission.

The first steps in the system development process involve defining the requirements for a new system and analyzing those requirements. The ultimate objective is to produce the system specification.

The conceptual analysis starts with a conceptual design of a system, which is based on applying a possible solution to the new system requirements. For automated systems the conceptual design often starts with the allocation of functions in a system to computers. This is done in a functional specification which is often an informal document (ROC -- Requirements of Operational Capability).

Many critical questions about the new system surface during conceptual analysis. Requirements are analyzed for consistency and completeness. What-if questions are posed to determine if requirements are attainable. In this phase, the analyst needs a tool which can provide quantitative answers to questions about the conceptual system design.

Simulation modelling is a technique which can provide a tool to give the analyst these answers. Simulation is based on the fact that models can be built which represent a real system. Responses to experiments conducted on the model are indicative of responses to similar conditions in a real system. This provides the performance and cost data that an analyst needs to analyze a conceptual design.

The goal is to define the requirements for the system which are consistent and attainable.

INTERACTIVE SIMULATION BY SYSTEMS ANALYST

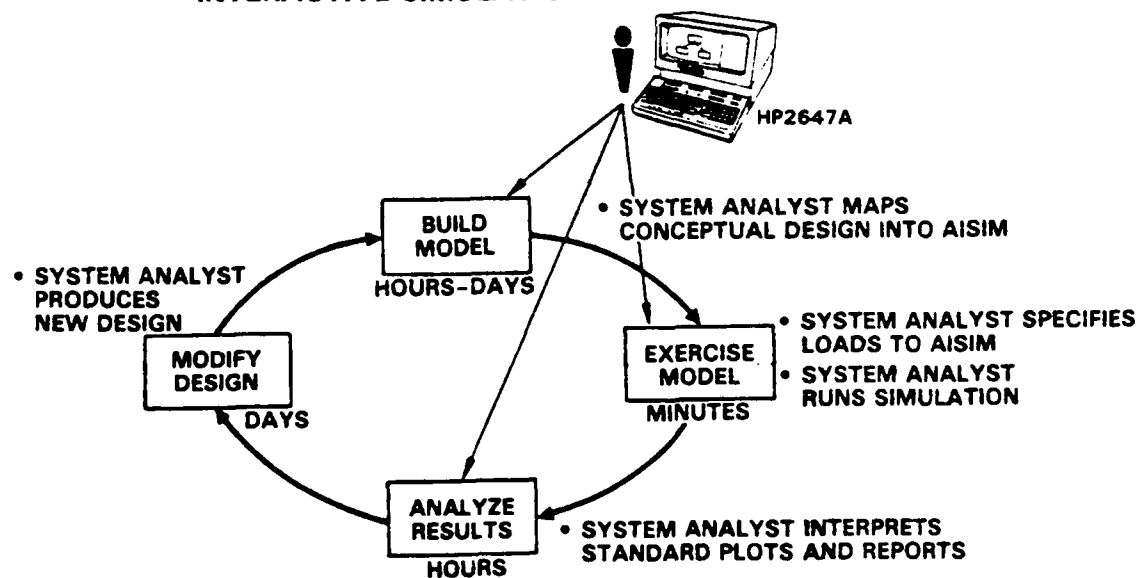


Figure 2. AISIM Simulation Process

AISIM is an interactive system which is used by the systems analyst to help define system requirements. It is used by the systems analyst to model a conceptual design. By using an interactive system to build models, exercise models, and analyze results, many questions about a system's functional specifications can be resolved in a timely and cost effective manner.

AISIM has been designed and built to be used by the systems analyst. The profile of this user is much different than that of a programmer. It is assumed that the systems analyst is not a frequent computer user. He does not know about computer system utilities like text editors or compilers nor does he want to learn. This implies that the AISIM user interface has to be a simple, powerful one, which is easily learned. The functions of AISIM support rapid model building, model running and model analysis.

Model building, model running and model analysis are the three key activities in the simulation process. Model building is the mapping of a system into a simulation model. Model running is the exercising of the model under some test conditions. Model analysis is the translation of the simulation results into meaningful statements about the system to support decision making.

2.2 Inputs to the AISIM Simulation Process

The input to a simulation effort comes from the functional specification of a system (ROC). The functional specification is the informal statement of the system requirements which is derived by applying a solution to a mission concept. A mission concept is what needs to be done. This has specific meaning in the context of military operations. A military operation is defined as the functions necessary to respond to threats and provide defense. A solution is the allocation and integration of functions in the operation so that the mission will be accomplished.

AISIM is directed at providing a tool to analyze systems in which computers are used to perform some of the functions. This is in the context of performing a mission.

2.2.1 Mission Concept and Requirements The mission concept is a description of the functionality of the system. The mission concept makes clear who the users of the system are and what the system does for them.

The mission requirements are the performance and loading characteristics of the environment in which the system satisfying the mission concept operates.

2.2.2 Problem Perspective The problem perspective identifies the "expected" areas of concern in the system and refers to the particular viewpoint an analyst takes in looking at a system. The perspective is derived by the analyst and generally focusses on the critical thread of processing for a mission.

2.2.3 System Description The system description is the conceptual system design which is proposed to satisfy the mission requirements. This often includes a system "sketch" identifying the major components of the system. Performance characteristics of the major components are also included in this description.

2.3 Preliminary Analysis

A preliminary analysis of the system is done to determine if discrete event simulation is applicable to the problem and to justify the use of AISIM. This is a screening process and usually is done very quickly. The preliminary analysis identifies the solution approach to analyzing the system. It is important to document the decisions made early in the analysis effort, which is the objective of this step.

2.3.1 Justifying AISIM Simulation The first step in AISIM simulation analysis is to justify the use of a discrete event simulation for the analysis of the problem. This is not too difficult for an AISIM user because discrete event simulation is one of the most powerful analysis techniques available. Using AISIM eliminates much of the risk associated with using this technique because it enables users to build models quickly for systems which have characteristics applicable to AISIM. To determine if AISIM is applicable, one looks at the system description and determines if the system has any of the following characteristics:

1. Procedural Operations - Processing in the system is described by a sequence of steps.
2. Parallel Processing - Any number of processing steps can occur simultaneously.
3. Resource Sharing - Elements of the system are shared. There is a discipline associated with sharing which governs the use of a resource.
4. External Loading - Activity in the system can be initiated by environmental stresses.
5. Interconnected Network Communication - Activities in the system communicate through defined channels.

Systems with these characteristics are easily modelled with AISIM.

The next step in justifying the use of AISIM is to determine if it is the best choice of available tools and techniques.

Considerations for this determination are the following:

Reasons for using AISIM

1. The AISIM user is a systems analyst. AISIM can be learned very quickly. Features of AISIM enable the user to build a model in a friendly environment.
2. AISIM enables the user to build models quickly.
3. AISIM provides automatic standard model documentation.
4. AISIM models are easily integrated with other AISIM models. AISIM allows models to be merged together or saved in a library.
5. AISIM has specific features for network modelling.

Reasons for NOT using AISIM

1. AISIM requires the use of a HP2647A terminal used in an interactive mode.
2. AISIM models are built from a high level description of a system. In order to model a system at a very low level some of the power of AISIM is neutralized.
3. AISIM models can only be run on computers on which AISIM is hosted.

Deciding whether AISIM is appropriate usually is done based on these types of considerations. Benchmarks conducted on AISIM show that users of AISIM can be trained to be expert users in a matter of days. Models done using AISIM have been shown to be built in substantially less time than doing the same model in a simulation programming language.

2.3.2 Define the Problem and Objectives Having decided that simulation is appropriate to the analysis of a conceptual design and that the input data described in section 2.2 is obtainable, the first activity which an AISIM user performs is to define the problem and state the objectives of the modelling effort. There are three steps in this which produce a statement of the problem with a solution approach. It is important that the statement be written as a document to provide visibility so that both the AISIM analyst and model reviewers know what is to be determined by the AISIM modelling effort.

2.3.2.1 Defining the Problem An AISIM model is a simplified abstraction of a system. Many elements and components of the system will be ignored in the model. In defining the problem, the AISIM analyst states the "expected" areas of concern found in the problem perspective, including those areas which will be addressed in the model and those which will not. For the omitted areas, justification should be given why they are not to be addressed. In defining the problem, the AISIM user documents any assumptions which are made about the systems mission. The assumptions, of course, should be consistent with the mission concept. Often, the assumptions address "holes" in the mission concept.

The boundaries of a model refer to the level of detail of the AISIM model to represent the elements both within the system and external to the system. In determining the boundaries of the model, the AISIM user makes a preliminary pass at mapping the components of the system into simulation entities. Specifically, it should be determined which components of the system are to be modelled by mathematical functions and which are to be modelled by AISIM entities. Also, the determination of what elements of the system will be modelled as Loads and what elements will be modelled by more detailed structures like Processes should be determined.

2.3.2.2 Defining the Simulation Objectives A clear statement of the objectives of the AISIM model should be written. This statement includes the questions about the system which the AISIM model will answer. The statement of objective provides visibility for what the simulation will accomplish.

2.4 Design, Plan and Construct the Model

Designing and planning a model are the activities a modeller performs to come up with a scheme for implementing a model of a system. A model design is the "document" which a modeller produces which identifies the mapping of components of a system into or onto components of a modelling tool. A model plan is the time sequence steps performed to implement the model. Constructing the model is the activity the modeller performs to implement the mapping using the modelling tool.

The model design lays out the model structure. The model structure shows the logical allocation of functions in the system in relation to functions in the model. This structure should look similar to the system structure in the system description with one exception. The model structure includes a representation of the system environment as a function. That is, the load generating part of the model is shown.

The first step in designing, planning and constructing a model is to determine the model structure and use that as the initial model design.

Designing, planning and constructing a model are activities which are done iteratively. This is not noticeably different than the plan, design and build steps in developing a system. When developing a system, though, it is very important to have a stable system design before starting to build components of the system. This is because the resources required to build components are usually costly so that it is desirable to use them efficiently. Using AISIM, a model can be constructed very quickly with very little expense. Therefore, it is possible to start the model building process before completing a model design with the knowledge that if part of the model is incorrect, it can easily be modified at some time in the future. The goal is to "get something running" which models some part of the system. This provides early results so that feedback on the modelling approach can be obtained as soon as possible. In conjunction with building an initial model prototype, a plan is developed for implementing the complete model of the system based on the model structure.

With the model structure a model design includes tables and charts which show how the components of a system map to simulation entities and how the simulation entities are related. The model plan describes how models of components of the complete system can be integrated into a complete system model. The model plan is produced as the model structure is determined. The plan is updated when necessary as a result of subsequent model design and construct activities.

The designing and building of AISIM models is an iterative process. First a component of the system is selected. A model is built of that component. This is called a submodel. The submodel is built and verified. Then another component of the system is selected. Each submodel is built and integrated. This continues until the complete model of the system is built which satisfies the problem definition. At each iteration it may be necessary to modify the submodels previously built to effect the integration.

It is expected at this step that an AISIM user will make use of submodels which have already been built and stored in the AISIM library. Since many systems have similar components, it is often possible to extract a submodel from another model and with only slight modifications, include it in a new model.

2.5 Exercise the Model

Exercising the model is the activity a modeller performs to stress the model system in order to obtain performance results. The model is exercised to first verify that it cycles correctly, and second to answer questions directed at the system conceptual design.

Using AISIM, a modeller describes the environment in which the

model system is to operate in terms of a Scenario. A Scenario definition should be carefully chosen. When verifying a model, the Scenario has to be easily understood. That means, it must generate a load on the system which can be traced, if necessary, to debug logic errors which have been included in the model by mistake. When supporting decisions about the system design, the Scenario must represent the actual environment in which the system is expected to operate.

Verifying the model is the activity a modeller performs to insure that the model which is built executes correctly. This corresponds to debugging a program and must be done to each sub-model as it is built. Normally, when one debugs a program, it is sufficient to test the single thread execution of the program through all of its threads of processing. Verifying a model or a submodel is somewhat more difficult than debugging a program because one has to test the time varying execution of the model in a multi-processing sense. This is done first by running a simulation on the model using simple Scenarios which stress the model in predictable ways. The model is verified by comparing results of the simulation with expected results. When results of the simulation do not match expected results, then there is either a problem with the model or the expected result is incorrect. A model is verified when a resolution between expected results and computed results for all verifying Scenarios is completed.

When verifying a simulation, it is advisable to eliminate all randomness from a model. This way it is possible to compute accurate expected results and define a Scenario which stresses the model in such a way to produce the results. There are two types of Scenarios generally used to verify AISIM models - single thread and multiple thread.

2.5.1 Single Thread Scenarios A single thread Scenario is one which exercises the sequential logic of the model without enabling resource contention or multi-processing logic. A single thread Scenario defines one path through the model logic. Results from running a simulation using a single thread Scenario should verify that all delay times and arithmetic computations are correct.

2.5.2 Multiple Thread Scenarios A multiple thread Scenario is one which exercises the resource contention and multi-processing logic of a model. A multiple thread Scenario defines many paths through the logic with simultaneous Process executions. A multiple thread Scenario should be set up to stress the model system in a very regular way so that expected results can be determined.

2.5.3 Analysis Scenarios Once the simulation has been verified so that a reasonable level of confidence exists that it is representative of the problem, analysis Scenarios are defined to determine the performance of the system under representative system loading conditions. The data for defining the system loading conditions is generally derived from the requirements of operational capability or from observation of existing systems performing similar missions. Associated with each analysis Scenario, the analyst should classify the Loads so that performance of the system produced by the simulation can be tracked to the Scenario.

2.6 Analyze Results

AISIM produces two standard outputs, interactive plots of the instantaneous simulation data and statistical summaries of all model entities. An AISIM user uses both of these to analyze the dynamics of the modelled system. Interactive plots of the simulation data are used to determine if the simulation reached a steady state, that is, no infinite queuing or other system bottleneck. If the system did not reach a steady state, then much of the data in the statistical summary can not be taken at face value; e.g., statistics on average performance are not valid. For each simulation run, a modeller should review all results and resolve any question about the system dynamics which is raised in his mind.

Analyzing simulation results is a challenging task. To do this well it is necessary to understand the Scenario stressing the model and how it relates to questions about the system design. Based on the Scenario, the analyst maps the results of the simulation experiment into concrete statements about the system performance in actual operation.

2.6.1 Identify Measured Performance Statistics AISIM outputs are standardized. This has the advantage that the AISIM user needn't concern himself with formatting reports or worry about the validity of computations. On the other hand, standardized reports may not provide the user with results in a form directly related to the wording of performance questions about the system. Because of this, it is necessary to identify the statistics in the AISIM statistical summary which are most relevant to the stated question.

2.6.2 Validate the Model Validation refers to determining whether a model accurately represents a system. Most AISIM models are validated through a review process, where the design and structure of the model are presented with results to people knowledgeable in the system modelled. The assumptions of the model should be described in detail. The model can be considered to be validated "by face" if no major objections or issues are raised in these reviews.

A simulation can be validated in part by analytic means. Resource utilization is the simplest calculation which can be done by hand and compared to simulation generated results. Resource utilization is a function of the "arrival rate" and the Resource "service time" calculated using queueing theory. The "arrival rate" corresponds closely to the mean time for Process triggering in an AISIM Load. The "service time" corresponds to Action delay times during which AISIM Resources are allocated. Resource utilization is reported under Resource # 3057 statistics in the AISIM Resource Report.

Comparing analytic results to simulation generated results provides an initial level of confidence of the validity of the model if results match closely. Plots of simulation generated data can support the simulation results reported in the AISIM Report. For this manual, the example problems will be validated by comparing results to analytic ones.

3. Example 1 - Communication Network

Example 1 is representative of a communication system consisting of many nodes which communicate through a subnet of switch processors. It is typical of the type of system addressed during the conceptual phase of command, control and communication system acquisition. The example demonstrates the use of AISIM to analyze a message-driven communication system consisting of many nodes with complicated routing strategies.

3.1 Input

3.1.1 Mission Concept

Many airbases are connected to headquarters through a communication network. The mission of the system is to provide communication between the bases and headquarters. The "users" of this system can be identified as the users wishing to communicate. Communication is implemented by electronic means using message switching processors connected in a subnet.

The message switching processors direct the data flow through the subnet and control the transmission of messages between nodes and across communication channels.

Airbases are the origin and destination of messages. Messages generated at airbases request functions to be performed at headquarters. Headquarters respond to requests by performing the requested function. In some cases data is returned to the airbases generating requests.

A command headquarters is the destination for all messages and is the source of display output messages.

3.1.2 Problem Perspective

A study of the following design elements is basic to the system to be modelled.

1. Network Topology - There are many different configurations of the hardware elements of this system. Depending on the physical distances and equipment used, there may be few or many subnet switch processors.
2. Channel Communications - Channel communications are described in terms of capacity (characters/sec) and protocol (half or full duplex).
3. Nodal Processing - Associated with the airbases and headquarters are processors which perform message generation and processing. Associated with subnet switch nodes are processors which perform message routing.

4. Message Types and Attributes - Messages are described by type (data display request, hardcopy display request, hardcopy data) each of which has the distinguishing attribute of length in characters.

The expected area of concern in this system focusses on the resource sharing of subnet processors and communication channels. The performance measures to be obtained from the model are the following:

1. Communications Queue Time - For each channel in the communication network, the time messages wait at subnet processing nodes to gain access to a communication channel.
2. Communications Channel Queue Length - For each channel, the number of messages waiting at subnet processing nodes for access to a communication channel.
3. Processor Queue Time - For each subnet node processor, the time messages wait to be routed or processed.
4. Processor Queue Length - For each subnet node processor, the number of messages waiting to be routed or processed.
5. Communication Channel Utilization - The average use of a communications channel for a unit time.
6. Processor Utilization - The average use of a processor for a unit time.

The critical thread is the sequence of events for transmitting process requests from source to destination nodes.

3.1.3 System Description The following description represents a communication system. The system is described by a proposed network design and a definition of the system traffic load.

3.1.3.1 Proposed Network Design The network is described by its topology, channel capacities, and routing tables. The suggested network topologies incorporate 3, 5, and 7 subnet nodes. Figure 3 illustrates these subnet topologies and shows the sources and destinations of message traffic. The three types of messages to be processed by this network are data messages, display request messages, and display output messages. The BS represent bases where messages may originate and terminate and where hardcopy display messages may terminate. The HQs represent headquarters which also originate and terminate message traffic and in addition have a graphic display capability.

The CHQ represents the command headquarters and is a destination for all messages, but it is only the source of display output messages. It is the only source of display output messages. That

is, it is never the source of data messages and it is always the source of display output messages. The CHQ contains a display processing capability which accepts a display request message, generates a display output message (hardcopy or graphic format) and sends it to the originator of the display request.

The Ss represent subnet nodes and perform only message switching functions. Ss are never the sources or destinations of any message traffic. The determination of the number and location of these subnet nodes is a major reason for the development of this model.

3.1.3.1.1 Message Flow Messages are transmitted in maximum size blocks of 4000 characters and flow from an originating B or HQ to its associated S, through the S network to the destination S, and finally to the destination B, HQ, or CHQ. Originating messages can be either data messages or display request messages. The data message flow is as just described. A display request message can originate at any B or at either HQ and terminate at the CHQ where the request is processed. A display output message is generated at the CHQ and the message is transmitted through the S network back to the originating B or HQ.

3.1.3.1.2 Routing Routing through the network will be in accordance with the tables shown in Table 2 for the three proposed network topologies. The coordinates in each table are the current node and the destination node. The intersection of these ordinates contains the number of the next node to which the message is to be transmitted.

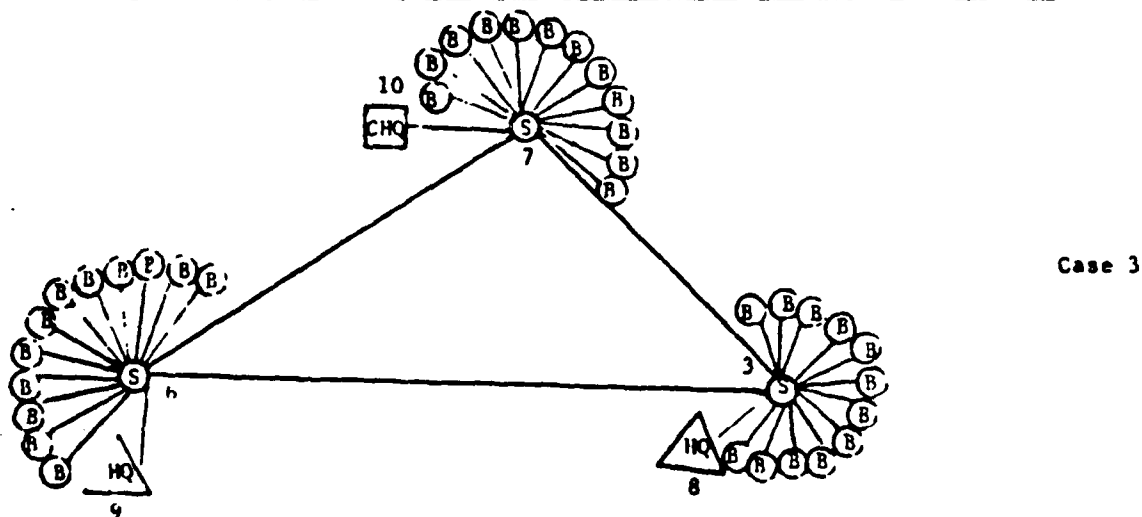
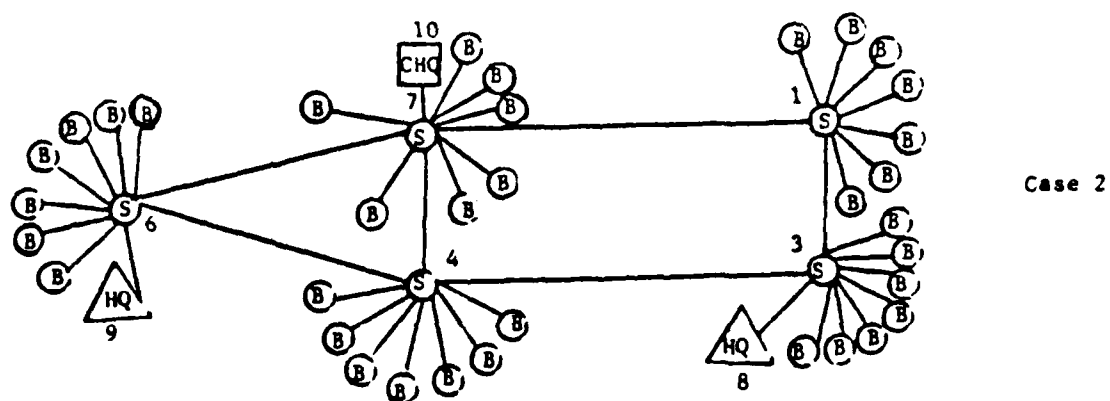
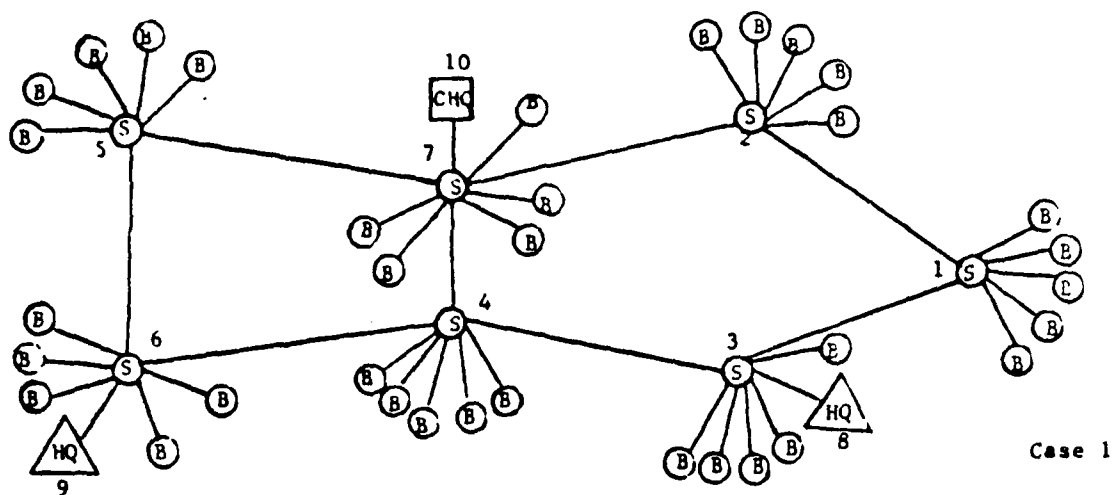


Figure 3. Example 1 Suggested Topology

3.1.3.1.3 Channel Capacities The effect of different channel capacities is to be examined in detail for the case 1 network topology. Figure 5 gives the different channel transmission rates for each analysis run. It is assumed that all channels are full duplex which means that they can transmit information in both directions simultaneously at the specified bit rates.

3.1.3.1.4 Nodal Processing Each node in the network has a processor associated with it to perform its message switching functions. The network model should incorporate delays for processing (both queueing time and processor time) at the S, H₂, and CH₂ nodes. Figure 5 contains character processing rates for the various nodal processors. Storage for each processor can initially be assumed to be 1,000,000 bytes but should be a variable which can be adjusted as necessary.

Routing

10	7	7	7	7	7	7	7	7	7	10
9	6	6	6	6	6	6	6	6	9	6
8	3	3	3	3	3	3	3	8	3	3
7	2	2	4	4	5	4	7	4	4	10
6	4	4	4	4	5	6	4	4	9	4
5	7	7	7	7	5	6	7	7	6	7
4	3	7	3	4	7	6	7	3	6	7
3	1	1	3	4	4	4	4	8	4	4
2	1	2	1	7	7	7	7	1	7	7
1	1	2	3	3	2	3	2	3	3	2
	1	2	3	4	5	6	7	8	9	10

Case 1

10	7	7	7	7	7	7	7	10
9	6	6	6	6	6	6	9	6
8	3	3	3	3	3	8	3	3
7	1	4	4	6	7	4	6	10
6	7	4	4	6	7	4	9	7
4	3	3	4	6	7	3	6	1
3	1	3	4	4	4	8	4	4
1	1	3	3	7	7	3	7	7
	1	3	4	6	7	8	9	10

Case 2

10	7	7	7	7	7	10
9	6	6	6	6	9	6
8	3	3	3	8	3	3
7	3	6	7	3	6	10
6	3	6	7	3	9	7
3	3	6	7	8	6	7
	3	6	7	8	9	10

Case 3

Figure 4. Example 1 Routing Tables

Channel Capacities

Case	Run	Data Compr. ^a	S - S Channels (kb/s)	S - HQ Channels (kb/s)	S - CHQ Channels (kb/s)	S - B Channels (kb/s)
1	A	No	19.2 ^b	40.8	230.4	4.8
	B	Yes	19.2 ^b	40.8	230.4	4.8
	C	Yes	9.6 ^c	9.6	230.4	9.6
	D	Yes	9.6	9.6	230.4	9.6
	E	Yes	9.6	9.6	230.4	4.8
2		Yes	9.6	9.6	230.4	9.6
3		Yes	9.6	9.6	230.4	9.6

^a Data Compression factor is 50 percent.

^b All S - S Channels except 4 - 7, which is 40.8 kb/s.

^c All S - S Channels except 4 - 7, which is 19.2 kb/s.

Figure 5. Example 1 Channel Capacities

Processor Variables

Processor	Processing Rate
S Processor	80 μ s/char
HQ Processor	80 μ s/char
CHQ Display Processor	100 μ s/char (hardcopy)
CHQ Display Processor	280 μ s/char (graphic)

Figure 6. Example 1 Processor Variables

3.1.3.2 Message Traffic Message input traffic for the network model consists of data messages and display request messages. The characteristics of these messages, i.e., length, input or output rate, and number of destinations are given in the Message

Traffic Characteristics Table, Figure 7. Messages are to be statistically generated by the model with exponential inter-arrival times (i.e., a Poisson arrival pattern). The message input rate shown is a total figure from all sources and is distributed randomly by source as shown in Figure 7. Message destinations are selected according to the traffic matrices presented in Figure 8. Data messages each have three destinations which differ according to source node as shown in the data message traffic matrix. An "X" in the matrix indicates a destination for that message source. The two 1/2s indicate that the messages are to be split equally between these two destinations. That is, for all data messages originating from a B connected to S7, one copy will always go to another B connected to S7, one copy will always go to CQ-10, and the third copy will alternately be delivered to HQ-8 or HQ-9.

Display request messages may originate from an HQ or a B connected to any S and that they are always addressed to the CQ (see Figure 3). Messages originating at the CQ are only display output messages generated in response to a display request and are addressed only to the B or HQ that originated the request.

Display output messages vary in length depending upon the type of display request--hardcopy or graphic. Messages destined for hardcopy printer output are 6300 characters in length while those being displayed on a graphic device are 10,000 characters in length. The time required for the processing of the request and the generation of the display must be simulated at the CQ. Figure 6 contains the processing rate variables for both of these functions.

The model routes messages over the pre-defined paths, creates message delays caused by line and node queues, and provides a statistical output of the specified network parameters. Section 2.2 illustrates the queueing and process functions which the model must produce. A message originates at a B, HQ, or CQ node where an output queue is established and the output queue time (Toq) is noted for the output channel. The transmission time (Tt) is accounted for to the S node, and a processor queue time (Tpq) and processor service time (Tp) are recorded. The message is then put on one or more output queues where the output queue time (Toq) and the transmission time (Tt) to the next node (B, S, HQ, or CQ) is calculated. No statistics are recorded for the final destination node with the exception of display request messages which go to the CQ and which cause the generation of a display output message. The time to generate this message must be accounted for in the CQ processor. The display output message is then treated as a new message which is placed on the output queue at the CQ destined for S7 and this message is processed through the network in the same manner as any message. However, the total message transit time for a display message will be the time from origin of the display request to delivery of the display output to the originator of the request.

Message Traffic Characteristics

Message Type		Mean Message Length ^b (8-bit Chars.)	Message Rate (Msg./Sec)	Number of Destinations ^e
Input Messages	Data Message	750	.517 ^c	3
	Display Request	200	.583 ^c	1
System Generated Outputs ^a	Display Output Hardcopy	6,300	.510 ^d	1
	Display Output Graphic	10,000	.073 ^d	1

^aGenerated in response to Display Requests.

^bExponential distribution. Length is without data compression. Maximum length per message partial is 4000 characters.

^cPoisson arrival pattern. Sources randomly distributed according to table I-6.

^dMessage destination generated according to sources specified in table I-6.

^eDestinations specified in traffic matrix tables.

Message Traffic Sources

Message Type	Distribution of Message Sources by Nodes (percent)			
	S1-7	HQ-8	HQ-9	CHQ-10
Data Messages	94	3	3	-
Display Request	82			
Hardcopy Output		2.5	2.5	
Graphic Output		6.5	6.5	

Figure 7. Message Traffic Characteristics

Data Message Traffic Matrix

Source		Destination									
		S ^b							HQ	CHQ	
		1	2	3	4	5	6	7	8	9	10
S ^a	1	X							X		X
	2		X						X		X
	3			X					X		X
	4				X					X	X
	5					X				X	X
	6						X			X	X
	7							X	$\frac{1}{2}$	$\frac{1}{2}$	X
HQ	8			X						X	X
	9						X		X		X

^a Subnet node to which Data Message originating node (B) is connected.

^b Subnet node to which Data Message destination node (B) is connected.

Display Message Traffic Matrix

Source		Destination									
		S ^b							HQ	CHQ	
		1	2	3	4	5	6	7	8	9	10
S ^a	1										X
	2										X
	3										X
	4										X
	5										X
	6										X
	7										X
HQ	8										X
	9										X
CHQ ^c	10	X	X	X	X	X	X	X	X	X	

^a Subnet node to which Display Request message originating node (B) is connected.

^b Subnet node to which Display Output message destination node (B) is connected.

^c Source of Display Response messages only.

Figure 3. Message Traffic Matrices

3.2 Preliminary Analysis

3.2.1 Justifying AISIM Simulation AISIM is applicable to this problem. The characteristics of the system map well into AISIM.

1. Procedural Operations - Messages originating at a source and routing through the network follow a sequence. The sequence is described by the following steps:

- Step 1 Message created at a source.
- Step 2 Message routed through network through subnet switch processors over communications channels.
- Step 3 Message received and processed at destination.
- Step 4 Response returned when appropriate.

This sequential operation is followed for all messages.

2. Parallel Processing - Any of the subnet nodes can be simultaneously handling messages.
3. Resource Sharing - Subnet nodes and communication channels are shared. A node may have many messages to process at any instant. Many messages may be ready to be transmitted over a communication channel.
4. External Loading - The loading on the network is described as message traffic characterized by a message rate in terms of messages per second. This is shown in the Message Traffic Characteristics Table, Figure 7. Messages are distributed by percentages over nodes. Each message is representative of a data processing request submitted by a user to the air base computer processor. The external loading on the communication is subtracted from the expected use of the airbase computer.
5. Interconnected Network - Different network configurations are posed as the key area of interest in this system.

3.2.2 Problem Definition

The problem definition for Example 1 can be stated as "representing the important elements of the described system by modelling entities". The selection of the elements of the system to be represented is made. This is described in the following statements.

1. All processors in the system will be represented. Processors will have characteristic attributes for computing processing delays based on the cycle speed of the processor and the number of characters processed. Processor use is governed by FIRST IN - FIRST OUT queueing logic.
2. All communication links in the system will be represented. Links will have the characteristics of communication channels for computing utilization of channels based on baud rate expressed in characters per second. Links will be governed by FIRST IN - FIRST OUT queueing logic.
3. The connectivity of the network will be represented by a matrix equivalent to the routing tables in the system description.
4. Message routing will be modeled for forwarding messages in the system from a source node to a destination node. The message forwarding will take into account the processing and queueing of messages through the network. Messages will be represented. Each message will have a separate instance for each occurrence.
5. Data compression will be represented by modifying the attributes of the instances of messages. This will be modeled by the AISIM Processes.
6. Different simulation runs will exercise the model according to the different cases of network topology and loading. A simulation will consist of an architecture for the system, a related legal path table for the architecture and a message loading modeled by the introduction of messages into the system at nodes over time.

The AISIM model of the communication network will address all the elements of the system described in the system description, with two noticeable exceptions. Since no airbase has different attributes than any other and the loads are specified according to S nodes, it is not necessary to model the airbase processors as 35 nodes. Rather the loads can be generated from one airbase node per S node, representing many airbases. There are two ways to do this.

One way is to specify that a number of resource units (seven for case 1) is to be associated with each airbase load node. Also each link between an airbase and a subnet switch S node can be considered to be seven channels. Using this scheme, the load for each airbase can be generated by increasing the message generation at a base by a factor of seven. This would not effect the loading on the subnet switch nodes and headquarters.

A second way to do this is to increase the processing capacities of the airbase processor and the associated channels, which

prevents queueing at the air base processor from effecting the load on the subnet switch nodes.

For the purposes of this model, the second alternative is sufficient. It is selected because it simplifies the model without invalidating it.

The second exception to the system definition involves the modeling of processor storage. Storage will not be considered because the requirements for storage are not adequately addressed in the system description. Storage could be viewed as the buffer storage requirements at each node for handling many messages. It could also include the processor storage for the programs which must perform the message processing as well as a storage management scheme. Since not enough data is available, it is best not to waste energy by including this in the model. Also, the requirements for buffer storage can be calculated from the queueing statistics of the processors and channels.

3.2.3 Definition of Objective

The objective of modelling the communications network is to produce quantifiable results for the system design for all performance measures stated in the system description. The different Scenarios and designs will be analyzed. Results will be tabulated and compared.

Of the various configurations described in section 3.1.3.1 only the seven subnet node network topology will be specifically addressed. The model will be built to enable rapid reconfiguration of the model to represent all three configurations. This will be done by making the logical Processes representing message routing independent of a specific architecture. The objective of this approach is to produce a general submodel for modelling message communication in any AISIM architecture.

3.3 Model Build

3.3.1 Design, Plan and Construction of the Model

3.3.1.1 Model Design The initial model design consists of a structure diagram showing the operation and interfaces of the model. The sequence of events which initiate activity in the model are described in this structure.

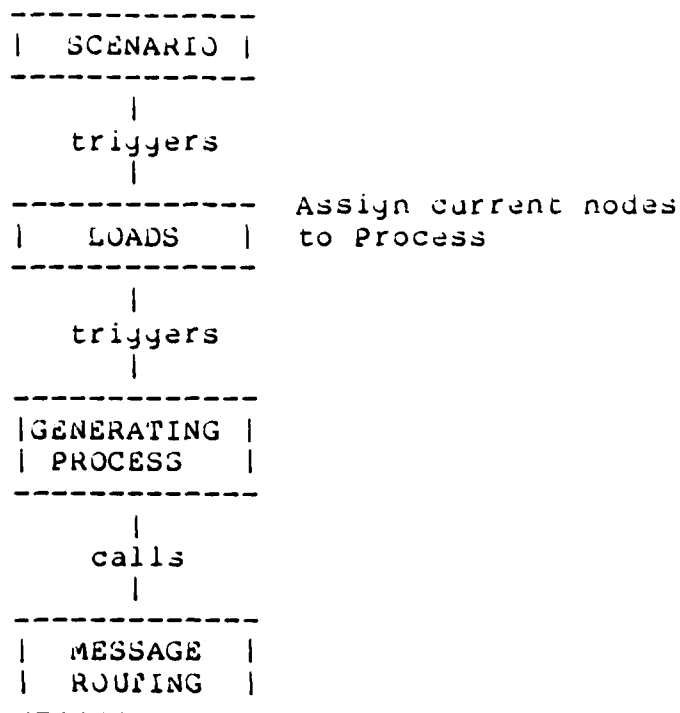


Figure 9. Example 1 Model Structure

3.3.1.2 Model Implementation Plan The sequence for constructing the model of the communication network is described in the following steps.

1. Design and Construct model for message routing.
2. Select and construct model of subset nodal processing.
3. Define subset loading Scenario.
4. Verify message routing on subset of network.
5. Build complete network architecture.
6. Include all nodal processing functions.
7. Define full loading.
8. Define full Scenario.

9. Verify complete network model.
10. Analyze results.

3.3.1.3 Model Construction

3.3.1.3.1 Message Routing Creating and routing messages through an architecture is the major technical feature of the system to model and so attention is first directed at the logic for the Processes which model this. The initial requirements for these Processes are derived from the formulation of the problem (section 3.2). The requirements are described below.

1. Messages are created with different attributes corresponding to origin, destination, length, requested processing and response option.
2. A Process exists in every node which can route a message received at a node to its destination. This Process can detect from the attributes of a message received at the node whether it is at its destination or not. If the message is at its destination, the processing requested by the message is initiated. If it is not, a channel transfer is initiated which forwards the message to its destination.
3. Channel transfers interrupt a node when a message has been transferred.
4. When a message reaches its destination and it is received and processed, a response is initiated if one is requested. A response message returns from the destination node to the origin of the message.

In order to meet these requirements for Processes, it is necessary to come up with a data and Process structure for the logic.

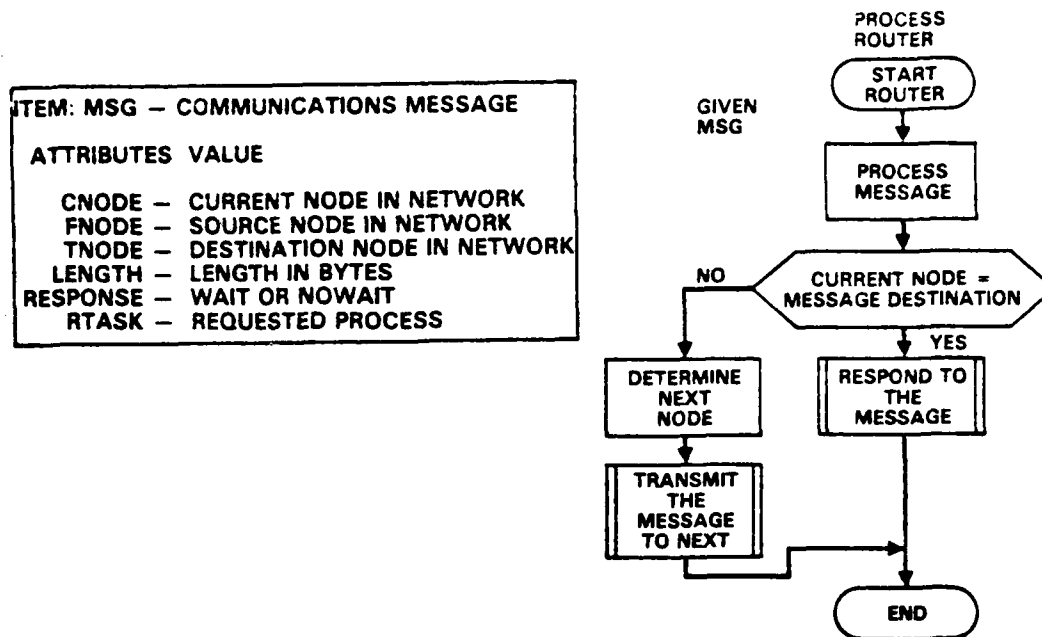


Figure 10. Message Routing Structures

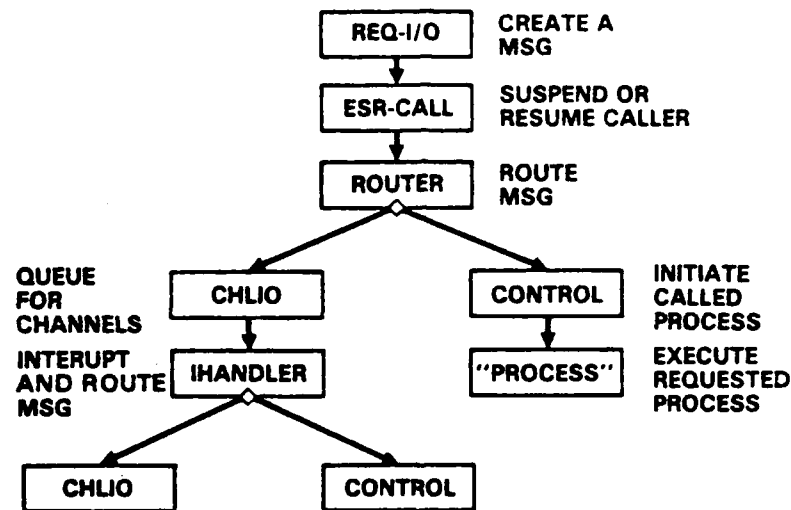
The Item, MSG, drives the logic of the routing Processes. "MSG" is routed through an Architecture by Processes. The Item must contain a number of attributes whose values provide the data which orients the Item in the Architecture at all instances of simulated time. Embedded in the attributes are the values for the source node of the Item, the destination node of the Item, the current node of the Item, the length of the communication message in bytes, the name of the requested Process to be initiated at the destination node and data on whether the Process which initiated the communication is waiting for a response or not.

The data on an Item "MSG" is used by the Processes which perform the message routing function. A key Process will be called ROUTER. The function of ROUTER is to determine if "MSG" is at its destination node or not. If "MSG" is at its destination node, the node responds to it by initiating the Process requested. If "MSG" is not at its destination node, the next node in the Architecture on the path to the destination node for "MSG" is determined and the Item is transmitted to it.

The sequence of the routing Processes is then developed. For each message, a similar sequence takes place. Names are given to the logical functions which correspond to each Process. REQ-I/O is the name of the Process which creates the communication message, "MSG", and assigns its attributes. "MSG" is passed to a

Process called ESR-CALL which determines if there is to be a response associated with the message. If there is to be a response, the initiating Process is suspended. If there is no response, the initiating Process is resumed. The next Process, ROUTER, determines the next node in the path to the destination node and initiates a channel transfer of the message to that node. CHLIO represents the queuing time for the channel and interrupts the next node. IHANDLER models the interrupt handling logic in the next node and continues routing the message through the Architecture until it reaches its destination node. When the message arrives at its destination node, the requested Process is initiated. If the initiating Process wants a response, a response message is created and routed back.

REQ-I/O INITIATES A CHAIN OF PROCESSES UTILIZING CHANNELS AND NODES ENROUTE



200000-20 0-12-021

Figure 11. Message Routing Process Sequence

3.3.1.3.2 Process: REQ-I/O

Process REQ-I/O is the top level Process of the Message Routing Submodel. This Process causes a Process request message to be generated. When this Process is called, it is given PROCESS, which is the name of a Process to be initiated in the destination node of the message, PRIORITY, which is the priority with which PROCESS is to be initiated, RESP.OPT, which is \$WAIT or \$NOWAIT and indicates whether the parent will wait until the initiated Process completes, MSG.LENGTH, which is the length in bytes of the message, TO.NODE, which is the destination node for the message and the node in which PROCESS is to be initiated. This Process does not return any parameters.

A detailed description of the parameters of this Process is given below.

PROCESS NAME: REQ-I/O - Generate a process request message on initiate I/O.

LOCATION: executes in all nodes.

GIVEN: PROCESS (DATA TYPE: PROCESS) - This Parameter is the name of the Process to be initiated in the destination node.

PRIORITY (DATA TYPE: REAL) - This parameter is the priority which the initiated Process is to have when it is started.

RESP.OPT (DATA TYPE: ALPHA) - This parameter is the option for the communication. The only legal values in this parameter are: \$WAIT - the parent Process will wait until the requested Process finishes before it resumes, \$NOWAIT - the parent Process does not wait on the requested Process.

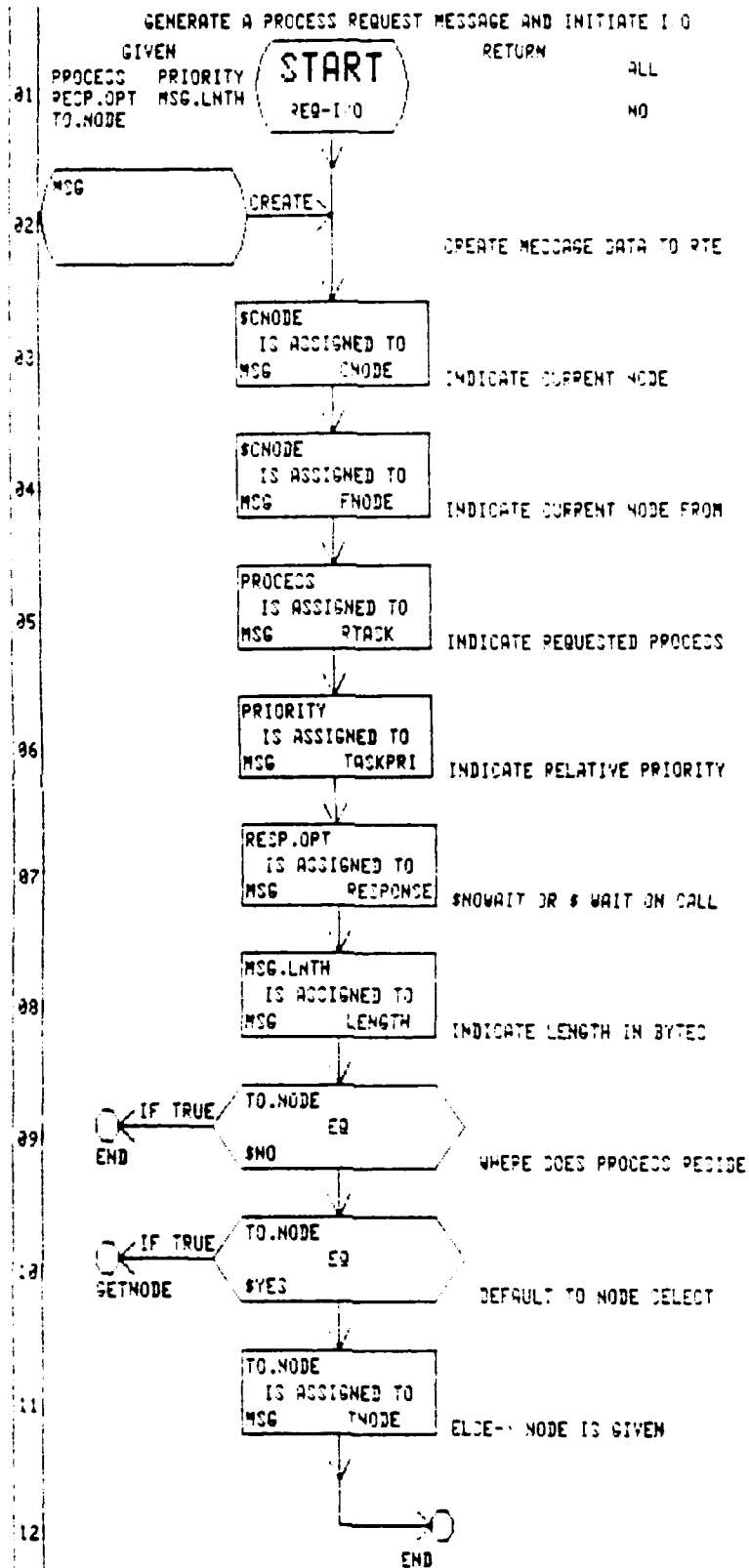
MSG.LENGTH (DATA TYPE: REAL) - This parameter is the length in bytes of the communication message routed through the network, requesting the Process to be invoked.

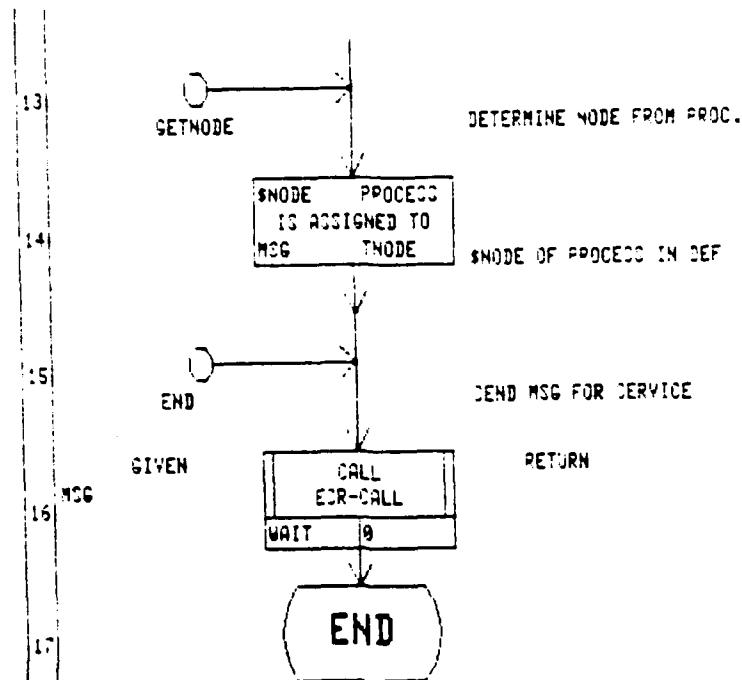
TO.NODE (DATA TYPE: RESOURCE or ALPHA) - This parameter is the destination node of the message which is the node in which to initiate the requested Process. If the ALPHA variable \$YES is provided for this parameter, the node of the requested Process is computed by the \$NODE keyword.

RETURN: NONE

CALLS: ESR-CALL

Following is the graphical representation of Process REQ-I/O.





REQ-I/O begins by creating the message and initializing various attributes of it. The attributes CNODE and FNODE are the current node in which this Process is executing. The attribute RTASK is set to the Process which will be executed in the destination node of the message. The attribute TASKPRI is set to the priority with which the requested Process will execute. The attribute RESPONSE is set to \$WAIT or \$NOWAIT; i.e. whether the parent is to wait for the requested Process to finish Processing. The attribute length is set to the length in bytes of the message. Next the value of the destination node is checked. If the value of the destination node is \$YES, then the TNODE attribute--i.e. the destination of the message--is set to be the node in which the requested Process executes. Otherwise, the name of the

destination node is supplied, and attribute FNODE is set to this value. This Process then calls Process ESR-CALL and gives it the created message.

3.3.1.3.3 Process: ESR-CALL

Process ESR-CALL is called by REQ-I/O and either suspends the requesting Process if a response message is requested (WAIT option) or allows it to continue processing if no response is requested (NOWAIT option). When this Process is called, it is passed the Item MSG. This Process does not return any parameters.

PROCESS NAME: ESR-CALL - Executive Service Request (CALL)

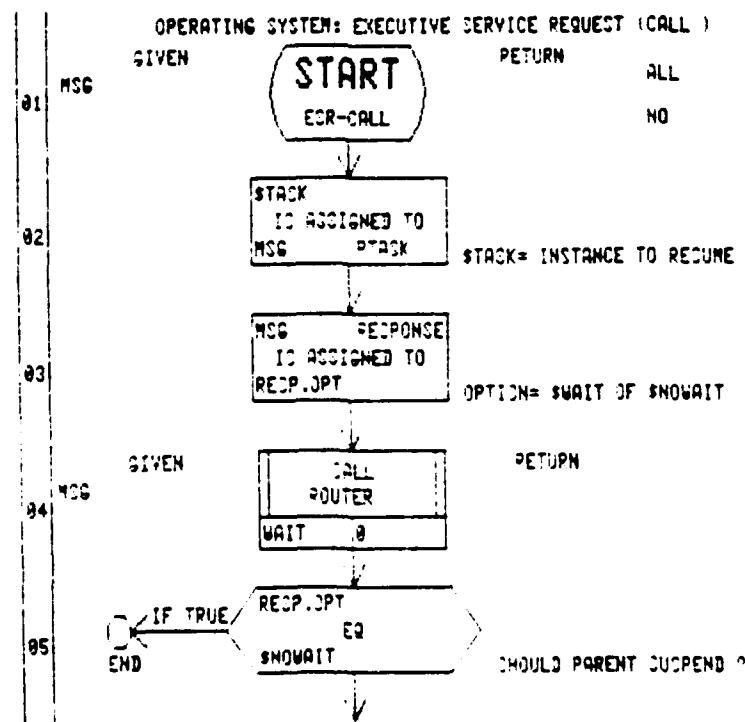
LOCATION: executes in all nodes

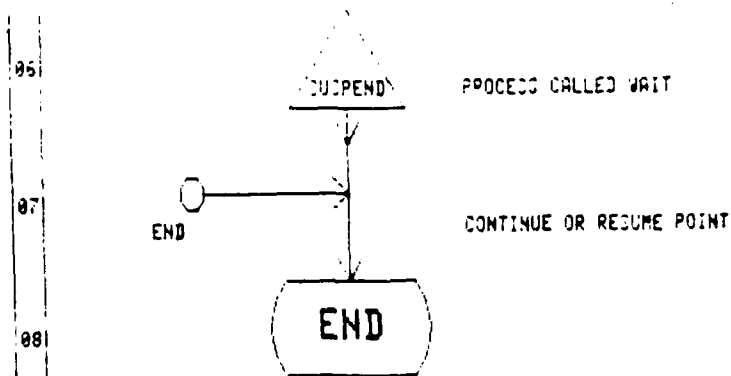
GIVEN: MSG (DATA TYPE: ITEM) - This parameter is the communication message created in REQ-I/O which contains the data for the Message Routing Submodel.

RETURN: N/A

CALLS: ROUTER

Following is the graphical representation of Process ESR-CALL.





This Process begins by setting the message attribute PASK to the currently executing instance of this Process. This value is maintained in case this Process is suspended and is to be resumed by another Process. Then the response option of the requesting Process for the requested option is retrieved. Then the Process calls the Process ROUTER to begin routing the message to its destination and waits for ROUTER to complete. Finally a test is made to see if the previously retrieved request option is \$WAIT or \$NOWAIT. If it is \$NOWAIT, ESR-CALL completes. If it is \$WAIT, ESR-CALL is suspended, having the effect that the requesting Process waits until the message reaches its destination, the requested Process executes, and a response is routed back before the requesting Process finishes Processing.

3.3.1.3.4 Process: ROUTER

Process ROUTER determines whether the message is at its destination node. When this Process is called, it is passed the message. This Process does not return any parameters.

PROCESS NAME: ROUTER - Operating System: Interrupt Handling and Routing

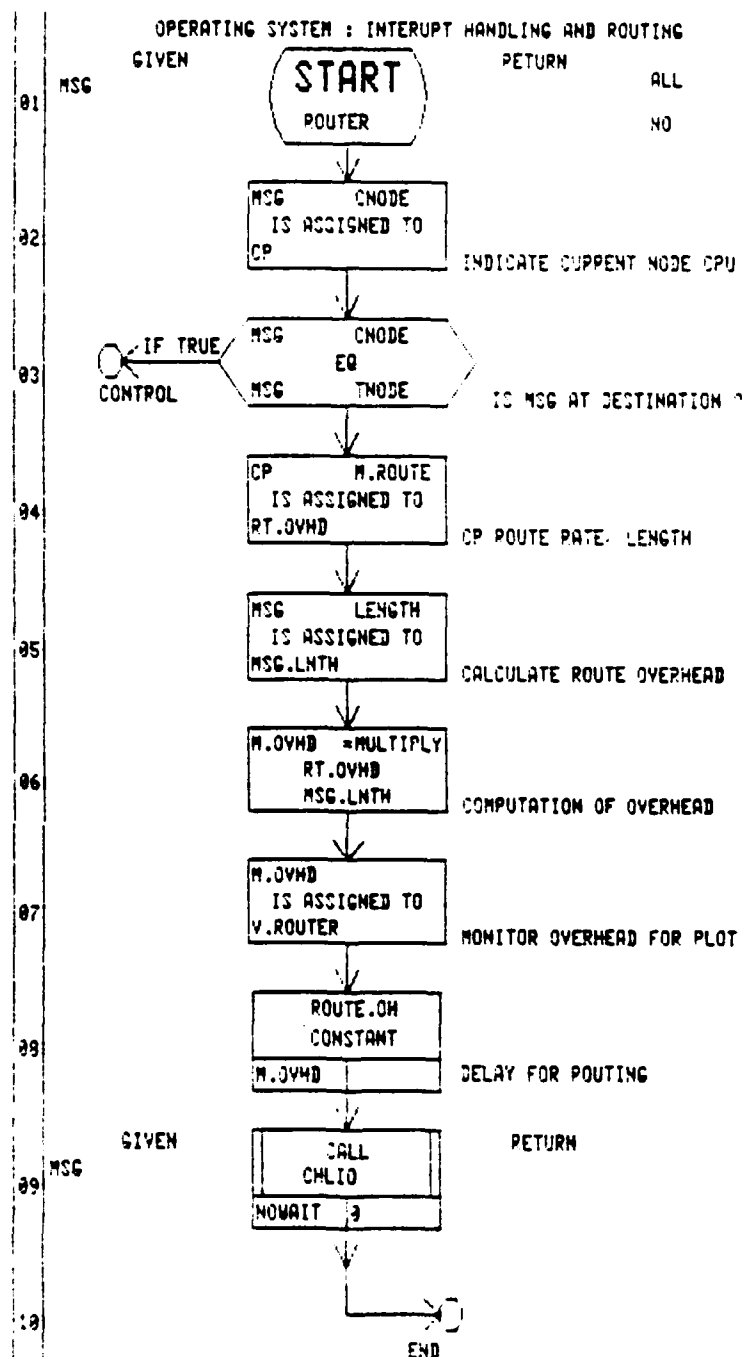
LOCATION: executes in all nodes

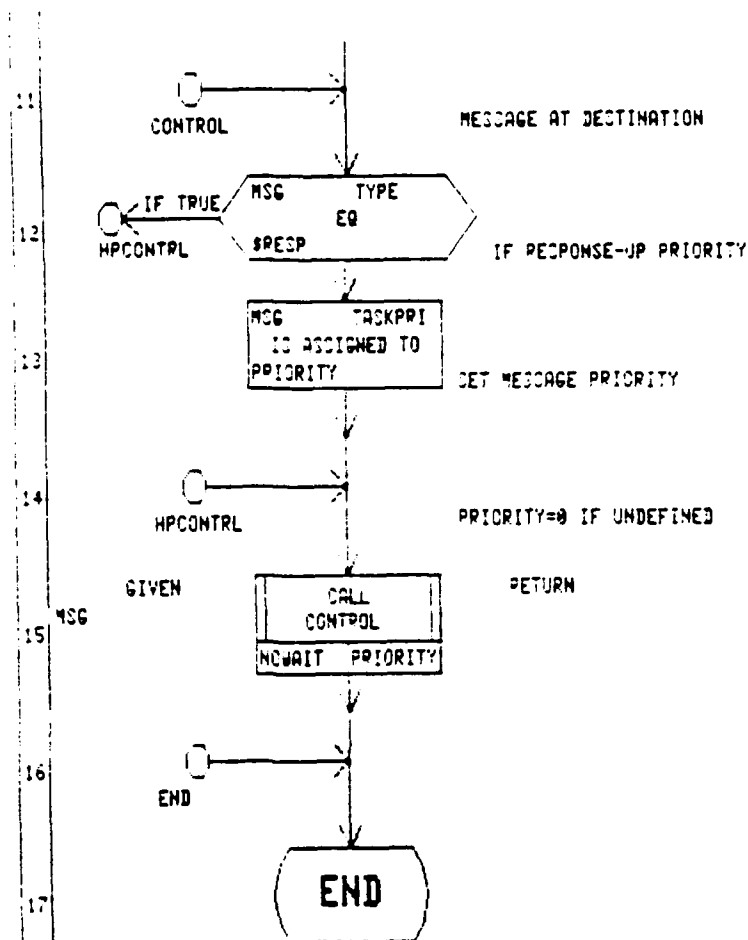
GIVEN: MSG (DATA TYPE: ITEM) - This parameter is the communication message created in REQ-I/O which contains the data for the logical process communication protocol.

RETURN: NONE

CALLS: CALIO, CONTROL

Following is the graphical representation of Process ROUTER.





The first step of this Process is to assign to a variable the name of the node which is the message's current position. The message's current position is then compared with its destination node. If at this point the node is at its destination, the destination node is the same node which generated the message. There is no routing overhead delay because the message did not need to be routed anywhere (i.e. M.CS is zero). The Process then tests to see if the message is a request message or a response message. If it is a request message, the routine CONTROL is called with a NOWAIT option and a priority equal to the requested priority, and the requested Process is initiated in the destination node. If the message is a response message, the routine CONTROL is called with a NOWAIT option and a priority of zero,

and the requesting Process is resumed, implying that the message has reached its destination, the requested Process has been initiated, and the message has been routed back to the node from which it was generated.

If the node is not at its destination, the overhead cost for transfer of the message from its current node to its next node must be calculated. Since this model is assuming that the overhead is a constant value because the messages are all the same length, the mean context switching time (M.CS) is used; the message length (MSG.LENGTH) and the route rate per length (RR.LENGTH) are not used. The Action ROUTE.ON is used to simulate this delay time. The routine CHLIO is then called NOWAIT to forward the message to its next node, and the Process terminates.

3.3.1.3.5 Process: CONTROL

Process CONTROL performs two different functions depending on when it is called. This Process is passed the message. If the message is a response message, then at this point the requesting Process has waited for the message to reach its destination, the requested Process to be initiated, and the message to be routed back. At this point the message has gone full circle and returned to the requesting Process's node. CONTROL then resumes the requesting Process. If the message is a request message, then the message is at its destination and the requested Process is initiated in that node. If the requesting Process is waiting for a response, then the attributes of the message are changed so that the original source node is now the destination node, and the message type is changed to a response type. Then the routine CHLIO is called to route the message back to the requesting Process's node. If the requesting Process is not waiting for a response, the message is destroyed.

PROCESS NAME: CONTROL - Operating System: Context Switching

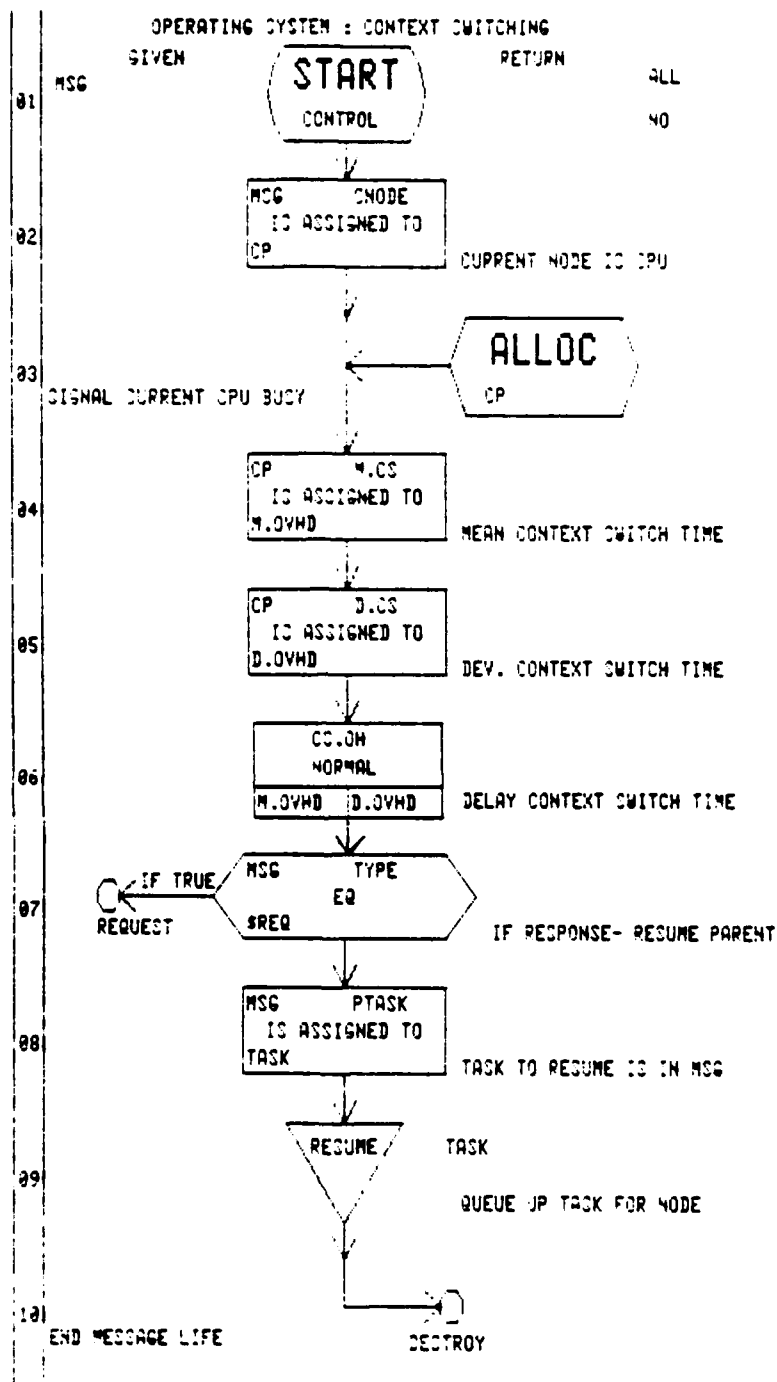
LOCATION: executes in all nodes

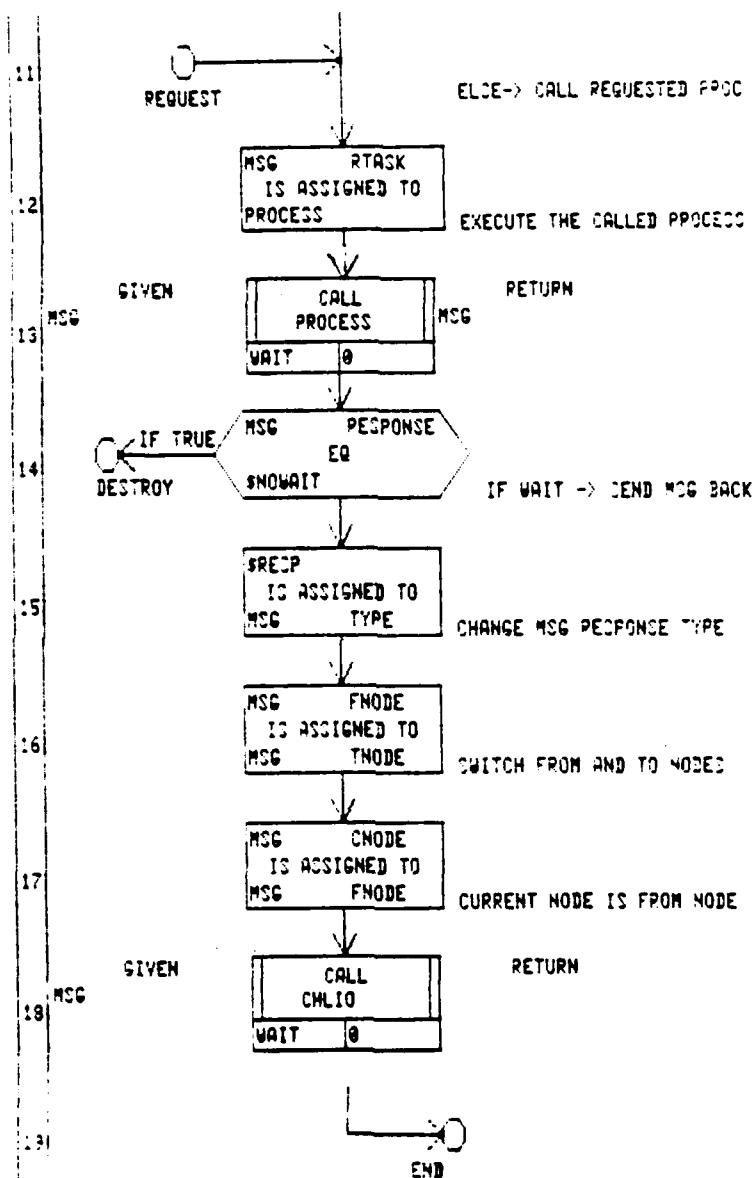
GIVEN: MSG (DATA TYPE: ITEM) - This parameter is the communication message created in REQ-I/O which contains the data for the logical process communication.

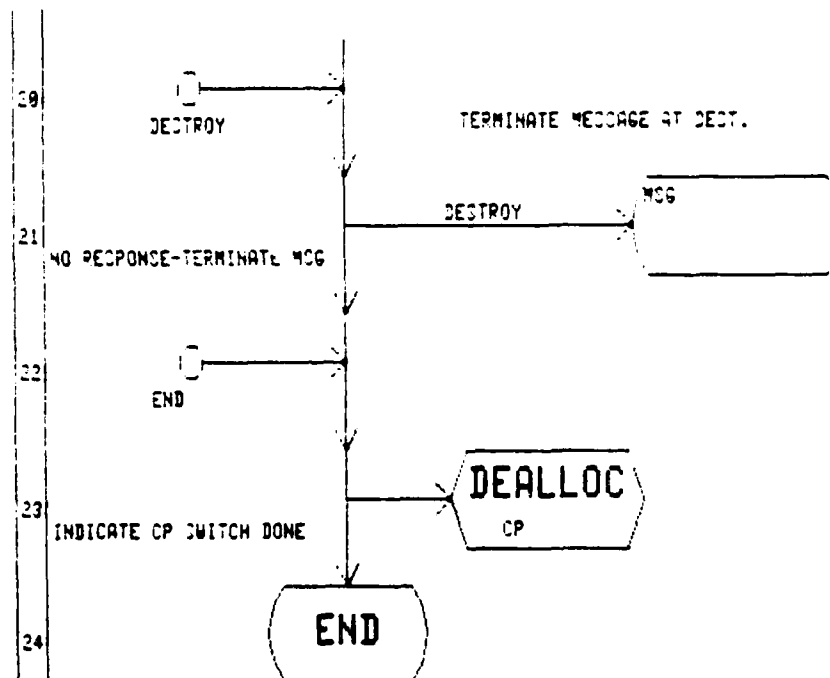
RETURN: NONE

CALLS: CHLIO

Following is the graphical representation of the Process CONTROL.







The first step is to allocate the node where the message is currently located. This is the node where the requested Process will be initiated or the node in which the requesting Process executes (which is currently suspended). The action CS.OH simulates the delay time involved in the context switching. The value for this time is an attribute of the node. Next the Process determines if the message is a request or response message. If it is a response message, this Process resumes the suspended instance of the requesting Process, destroys the message, deallocates the current node, and terminates. If the message is a request message, the name of the requested Process is retrieved from the PFASK attribute of the message and the Process is initiated. CONTROL waits until the requested Process completes.

Next CONTROL checks the message attribute RESPONSE to see if the requesting Process is waiting for a response. If a response is not desired, the message is destroyed and CONTROL terminates. If a response is requested, the message type is changed to response, the destination node is changed to the from node and the from node is changed to the current node. Then the Process ROUTER is called to route the message back to its origin. ROUTER is called with a WAIT option. CONTROL then terminates.

3.3.1.3.6 Process: CHLIO

Process CHLIO determines the current node and the destination node for the message which is passed to it. It then accesses the Legal Path Table to determine the next node along the route and the channel to get there. The channel is allocated to simulate its use, and the routine IHANDLER is called to interrupt the next node.

PROCESS NAME: CHLIO - full and half duplex channel logic

LOCATION: executes in all nodes

GIVEN: MSG (DATA TYPE: ITEM) - This parameter is the communication message created in REQ-I/O which contains the data for the logical process communication.

RETURN: NONE

CALLS: IHANDLER

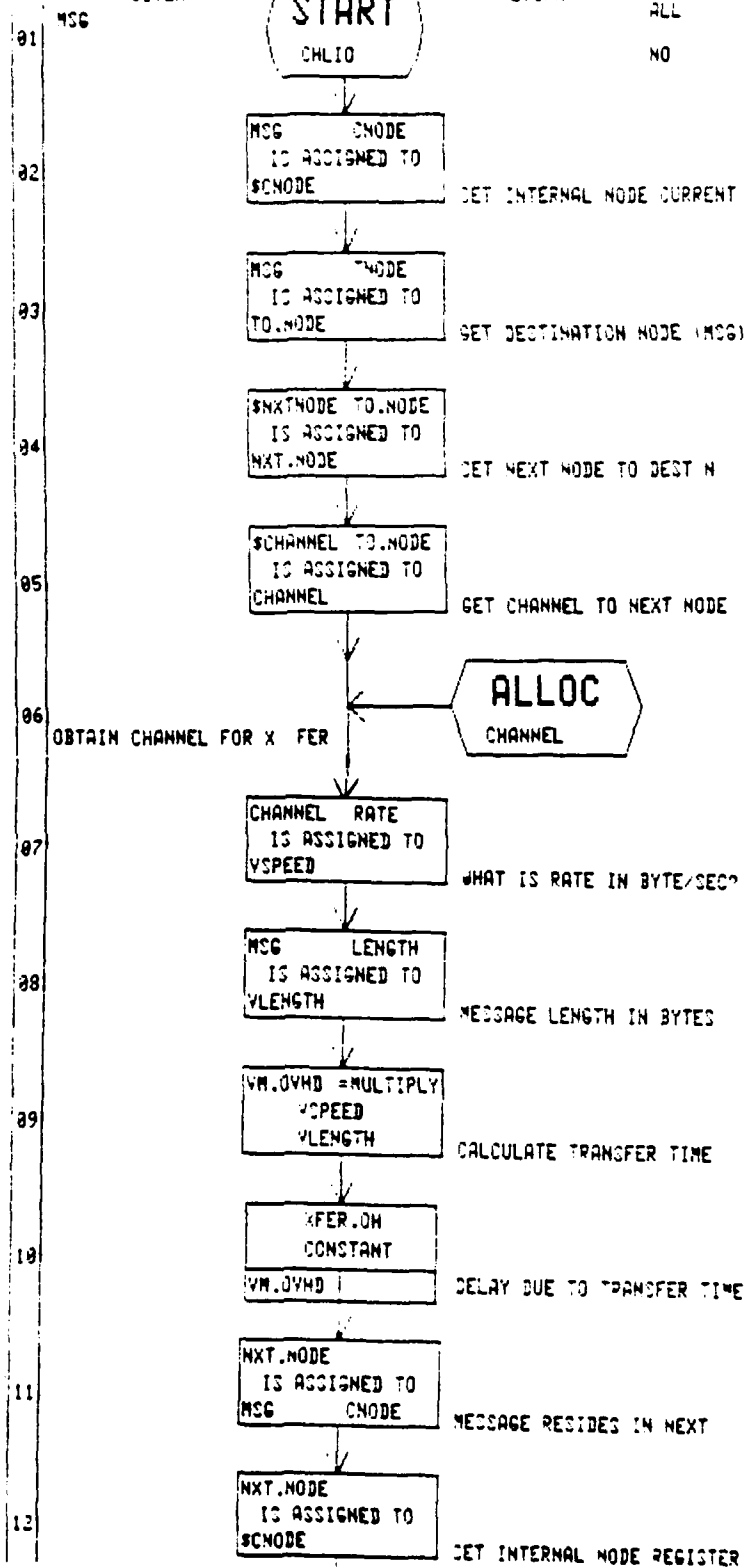
Following is the graphical representation of the Process CHLIO.

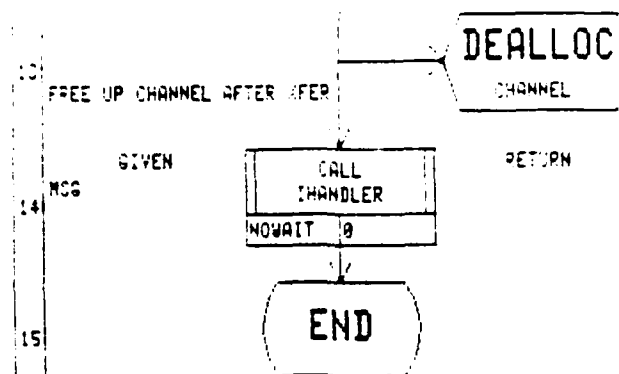
GIVEN

RETURN

ALL

NO





The first step of this Process is to assign the current node of the message to \$CNODE and to get the destination node for the message. Then the next node and link are determined based on \$CNODE and the destination node. Then the link is allocated. The transfer time for the message to cross the channel is always a constant rate (the EVAL calculation using the channel rate and the message length is ignored). The Action XFER.CH simulates the time used to traverse the channel. Then the current node attribute of the message is changed to the next node to update the message's position, and the current node (\$CNODE) is set to the next node. The link is then deallocated and the routine IHANDLER is called to interrupt the next node Processor.

3.3.1.3.7 Process: IHANDLER

Process IHANDLER is similar to the Process ROUTER. IHANDLER is passed the message. The Process then interrupts a processor node by allocating the node. If the message is not at its destination node, then IHANDLER computes the next node in the route to the destination node and calls the routine CHLIO to perform the routing. If the message is at its destination node, then IHANDLER calls CONTROL to initiate the requested Process in the destination node.

PROCESS NAME: IHANDLER - Operating System: Interrupt Handling and Routing

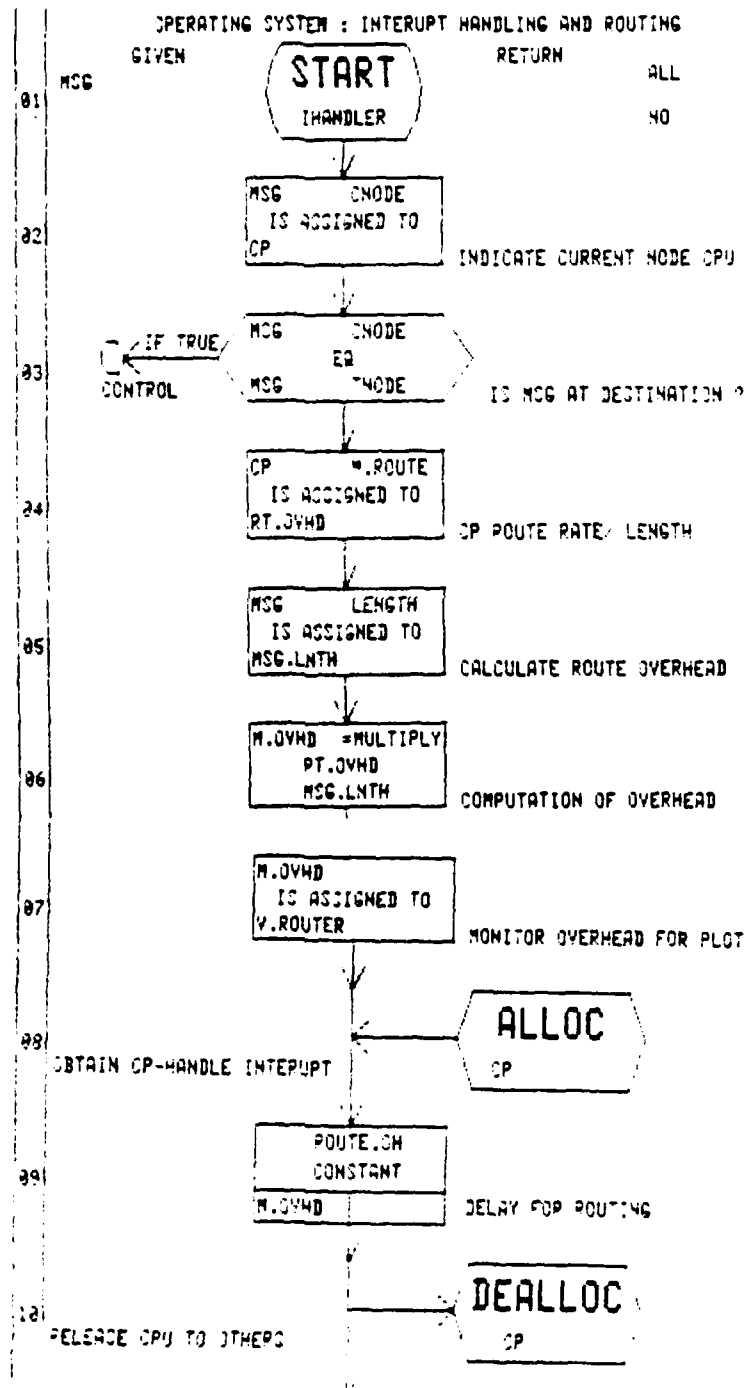
LOCATION: executes in all nodes

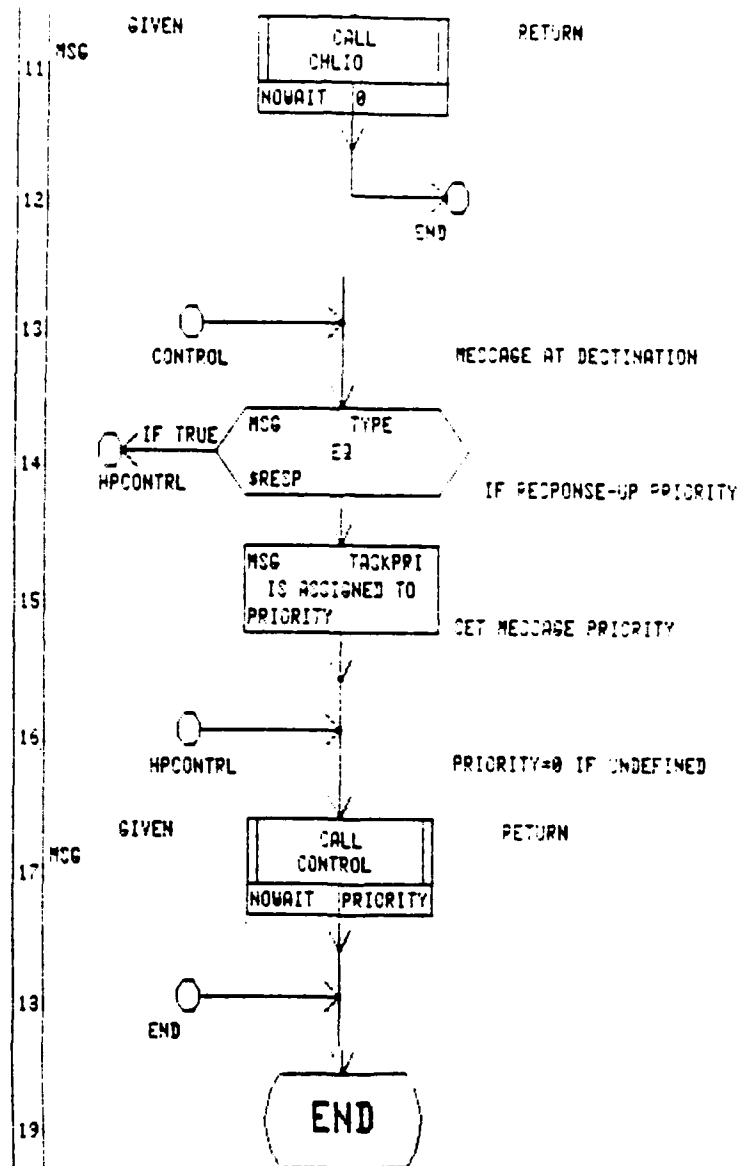
GIVEN: MSG (DATA TYPE: ITEM) - This parameter is the communication message created in REQ-I/O which contains the data for the logical process communication protocol.

RETURN: NONE

CALLS: CHLIO, CONTROL

Following is the graphical representation of Process IHANDLER.





This Process first determines whether the message is at its destination node. If it is, IHANDLER calls routine CONTROL to initiate the requested Process. If a response message is to be sent, then CONTROL is called with a zero priority; otherwise, it is called with the priority of the requested Process. Either way, CONTROL is called with a NOWAIT option. If the message is not at its destination, the current node is allocated. The Action ROUTE.OH is used to simulate the delay time for Processing the routing. Then the node is deallocated and routine CHLIO is called NOWAIT to perform the routing. IHANDLER then completes.

3.3.1.4 Subset of System A single thread Scenario exercising the Message Routing Submodel for a subset of the communication network is used to verify the logic of the submodel. The single thread Scenario uses three nodes and models a hardcopy request transmitted from airbase B7 to the command headquarters Cd2 through subnet switch processor S7. The simple architecture is shown below.

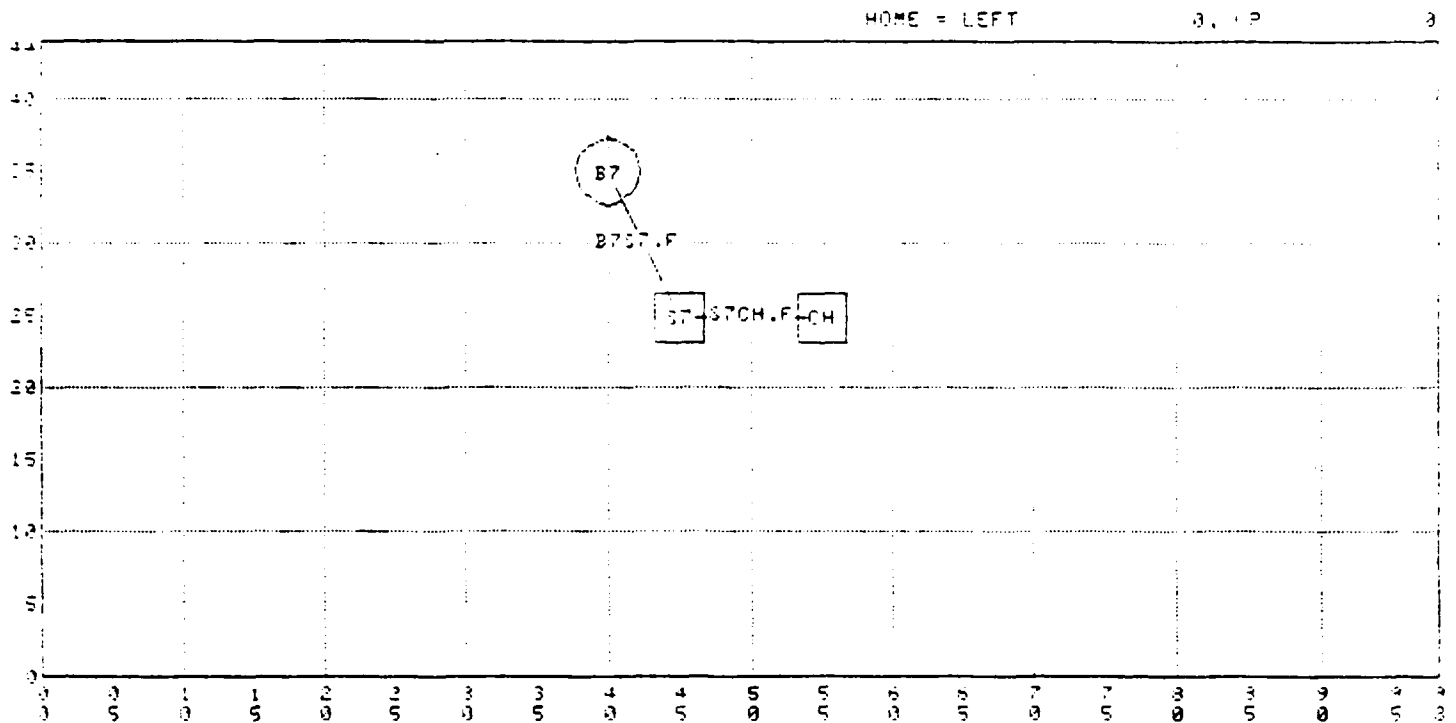


Figure 12. Example 1 Subset Architecture

The operation of this test case initiates with a Scenario TEST1 defined in the following way. TESTLOAD initiates the Process HCOFYCHQ at node B7 at the start of the simulation. This is accomplished with the following Scenario and Load definitions.

SCENARIO NAME: TEST1 PERIOD LENGTH: 600000

DESCRIPTION: SINGLE THREAD TEST CASE

PERIODS: 1

<u>TRIGGER</u>	<u>SCH TIME</u>	<u>PRIORITY</u>
TESTLOAD	0	0

LOAD NAME: TESTLOAD

NODE1
B7

DESCRIPTION:
INITIATE PROCESS HCOFYCH1

<u>PROCESS</u>	<u>MAX #</u>	<u>SCH MTD</u>	<u>MEAN</u>	<u>DELTA</u>	<u>PRIORITY</u>
HCOFYCH1	1	START			0

Figure 13. Example 1 Single Thread Test Load and Scenario

3.3.1.4.1 Process: HCOFYCHQ

The Process HCOFYCHQ represents the request for hardcopy from an airbase to the command headquarters. It has the effect of triggering through a CALL to REQ-I/O which creates a message, MSG, with LENGTH attribute equal to HCLNTH (200 characters). The TRACE Primitive enables the TRACE output which the AISIM simulator automatically generates.

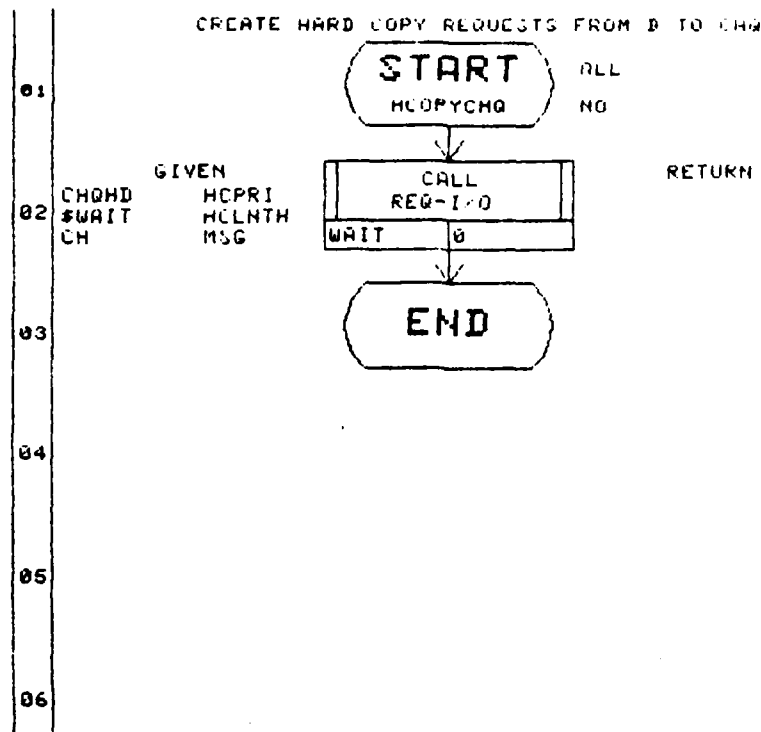


Figure 14. Process dCOPYCHQ

3.3.1.4.2 Process: CHQHD

The Process CHQHD is initiated in node CH after the communication message is routed through the network. Hardcopy graphic response messages are 6300 characters. The Process CHQHD represents the reception and processing of hardcopy requests from the two secondary headquarters HJ1 and HJ2. It is initiated by the Processes HJ1HGEN and HJ2HGEN through a CALL to REQ-I/O. The Process is given a reference to the Item MSG as a parameter. The first primitive, ASSIGN, sets the value of the attribute of MSG called dCRLNTH to the local variable MSG.LNTH representing the length of the hardcopy data message. In the EVAL primitive that follows, the local variable M.OVHD is set to the product of MSG-LNTH and CHQDOVHD, which is a global variable representing the overhead for hardcopy requests in bytes/second. The variable M.OVHD is thus equal to the time overhead of the response to the hardcopy request from HJ1 and HJ2. An appropriate amount of time is then taken up in an ACTION primitive and the Process then returns the reference to the Item MSG to the calling Process. The graphic representation of CHQHD is shown in Figure 15.

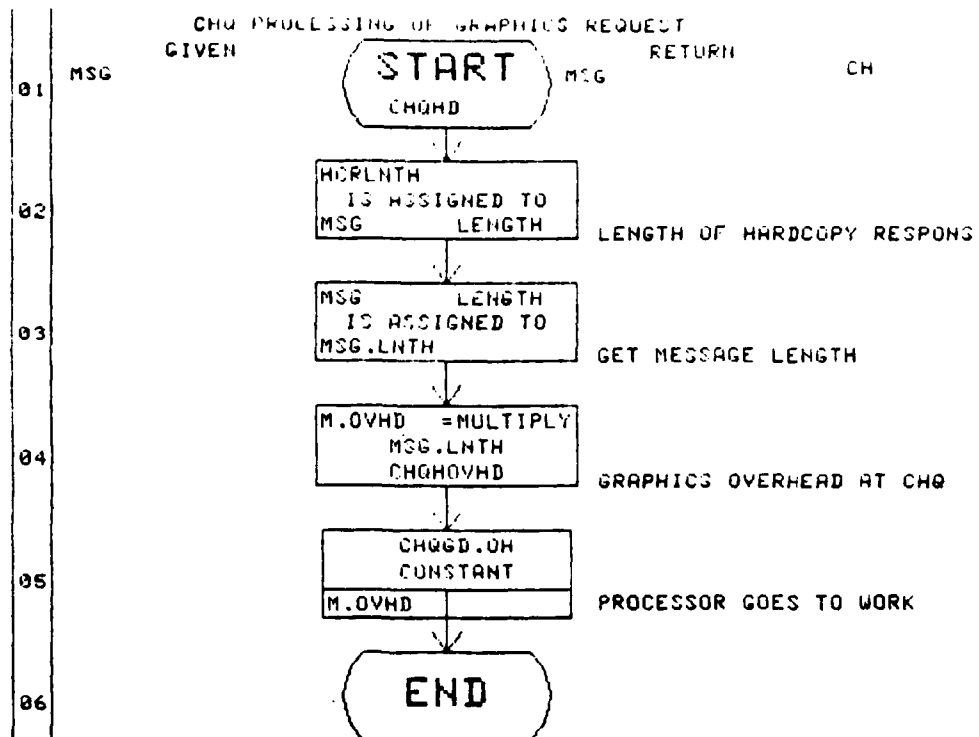


Figure 15. Process CHQHD

3.3.1.4.3 Single Thread Trace The Single Thread trace is shown below.

T=	0	N = B7	P = TRACE	TRACE ON
T=	0	N = B7	P = TRACE	END
T=	1600.00000	N = B7	P = ROUTER	CALL CHLIO
T=	1600.00000	N = B7	P = ROUTER	END
T=	1600.00000	N = B7	P = CHLIO	START
T=	1600.00000	N = B7	P = CHLIO	ALLOCATE B7S7 A
T=	1600.00000	N = B7	P = ESR-CALL	SUSPEND
T=	8259.99609	N = S7	P = CHLIO	DEALLOC B7S7 A
T=	8259.99609	N = S7	P = CHLIO	CALL IHANDLER
T=	8259.99609	N = S7	P = CHLIO	END
T=	8259.99609	N = S7	P = IHANDLER	START
T=	8259.99609	N = S7	P = IHANDLER	ALLOCATE S7
T=	8260.01209	N = S7	P = IHANDLER	DEALLOC S7
T=	8260.01209	N = S7	P = IHANDLER	CALL CHLIO
T=	8260.01209	N = S7	P = IHANDLER	END
T=	8260.01209	N = S7	P = CHLIO	START
T=	8260.01209	N = S7	P = CHLIO	ALLOCATE S7CH A
T=	14920.00819	N = CH	P = CHLIO	DEALLOC S7CH A
T=	14920.00819	N = CH	P = CHLIO	CALL IHANDLER
T=	14920.00819	N = CH	P = CHLIO	END
T=	14920.00819	N = CH	P = IHANDLER	START
T=	14920.00819	N = CH	P = IHANDLER	CALL CONTROL
T=	14920.00819	N = CH	P = IHANDLER	END
T=	14920.00819	N = CH	P = CONTROL	START
T=	14920.00819	N = CH	P = CONTROL	ALLOCATE CH
T=	14920.00821	N = CH	P = CONTROL	CALL CHQHD
T=	14920.00821	N = CH	P = CHQHD	START
T=	77920.00821	N = CH	P = CHQHD	END
T=	77920.00821	N = CH	P = CONTROL	CALL CHLIO
T=	77920.00821	N = CH	P = CHLIO	START
T=	77920.00821	N = CH	P = CHLIO	ALLOCATE S7CH B
T=	287709.88321	N = S7	P = CHLIO	DEALLOC S7CH B
T=	287709.88321	N = S7	P = CHLIO	CALL IHANDLER
T=	287709.88321	N = S7	P = CHLIO	END
T=	287709.88321	N = S7	P = IHANDLER	START
T=	287709.88321	N = S7	P = IHANDLER	ALLOCATE S7
T=	287709.88321	N = CH	P = CONTROL	DEALLOC CH
T=	287709.88321	N = CH	P = CONTROL	END
T=	287710.38721	N = S7	P = IHANDLER	DEALLOC S7
T=	287710.38721	N = S7	P = IHANDLER	CALL CHLIO
T=	287710.38721	N = S7	P = IHANDLER	END
T=	287710.38721	N = S7	P = CHLIO	START
T=	287710.38721	N = S7	P = CHLIO	ALLOCATE B7S7 B
T=	497500.26221	N = B7	P = CHLIO	DEALLOC B7S7 B
T=	497500.26221	N = B7	P = CHLIO	CALL IHANDLER
T=	497500.26221	N = B7	P = CHLIO	END
T=	497500.26221	N = B7	P = IHANDLER	START
T=	497500.26221	N = B7	P = IHANDLER	CALL CONTROL
T=	497500.26221	N = B7	P = IHANDLER	END
T=	497500.26221	N = B7	P = CONTROL	START
T=	497500.26221	N = B7	P = CONTROL	ALLOCATE B7
T=	497500.26221	N = B7	P = CONTROL	RESUME ESR-CALL
T=	497500.26221	N = B7	P = CONTROL	DEALLOC B7
T=	497500.26221	N = B7	P = ESR-CALL	ALLOCATE B7
T=	497500.26221	N = B7	P = CONTROL	END
T=	497500.26221	N = B7	P = ESR-CALL	END
T=	497500.26221	N = B7	P = REQ-I/O	END
T=	497500.26221	N = B7	P = HCOPIYCHQ	DEALLOC B7
T=	497500.26221	N = B7	P = HCOPIYCHQ	END

Figure 16. Example 1 Single Thread trace

3.3.1.5 Complete Network Architecture The specification requires a system of many airbases, two headquarters, a single command headquarters and switches to route messages between any of the other physical elements in the system. A number of architectural topologies would accommodate this description. The purpose of the model is to test switching capacity. Statistically speaking, each of the airbases imposes the same load on the system. A simplification to replace the collection of airbases around each switch by a single airbase, compensating for the reduced number by increasing the load from the single base by an appropriate factor, can be done. Adapting the 7 subnet architecture by this simplification produces an architecture for the system shown in Figure 17.

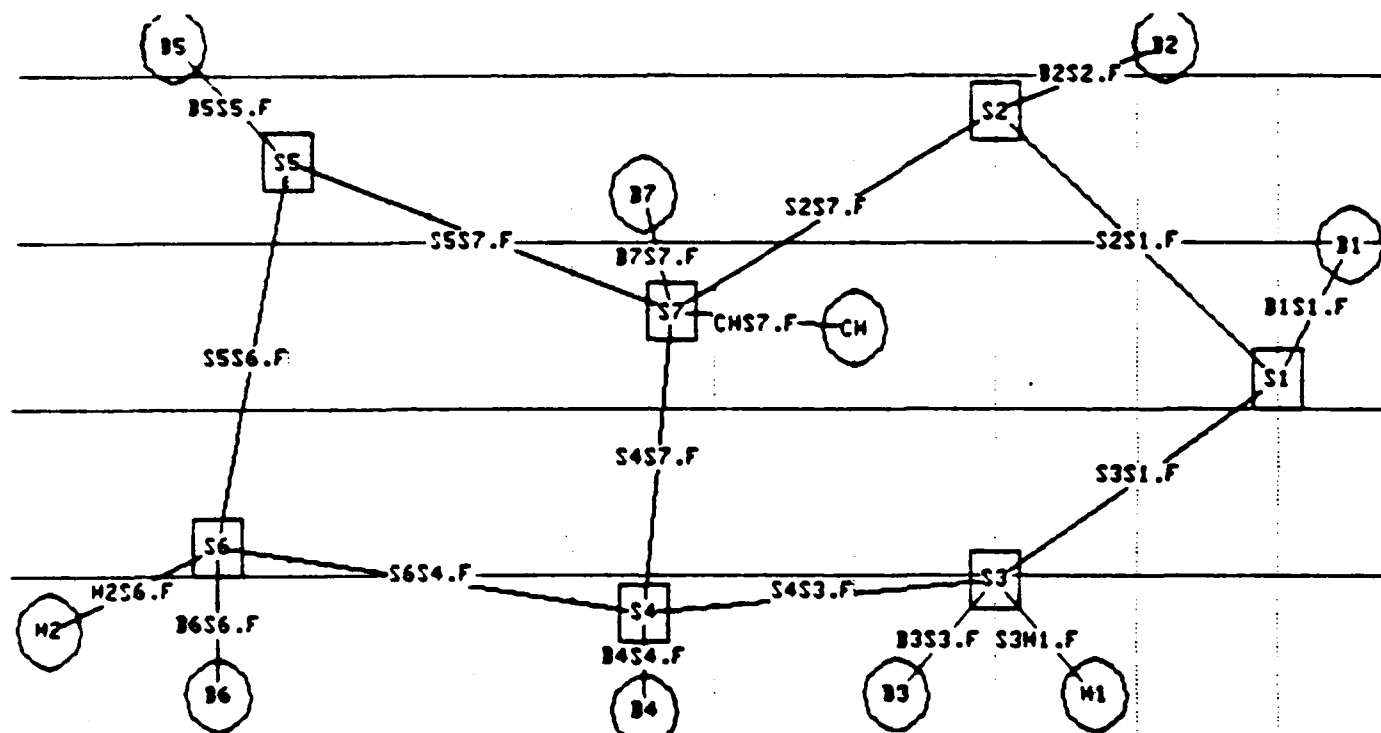


Figure 17. The Complete Network Architecture

where the six nodes labeled "B" represent airbases; the nodes labeled "H" are headquarters and the node "CH" represents a command headquarters; the squares labeled "S" are the switches through which any messages between airbases, headquarters or command headquarters must be routed. The Legal Path Table is in the main determined by the topology of the network itself, in conjunction with the routine tables (figure 4). Leaf-nodes representing airbases, headquarters and command headquarters communicate with any other nodes in the system through its contiguous switch. The architecture, through the presence of the link between the switches S4 and S7 (together with the question of direction), allows for several alternative paths between

switches, amongst which the user must choose explicitly.

3.3.1.6 Node Process Definitions In order to represent node processing functions, twenty-two distinct Processes representing the various operations of the communication system had to be defined. Figure 18 contains a table showing the system of Processes in the model. The left hand column indicates the name of each Process as it appears in the model. The middle column indicates which of the three activities modeled in the Process are represented by the named Process and the third column shows the Process that is triggered by the one named to the left and what activity it represents. The AISIM representation of each of the Processes is described with a rationale for its design.

MSG GENERATION PROCESS	GENERATION FUNCTION (MSG TYPE: TRANSMISSION)	PROCESS CALLED AT DESTINATION
DATABB01...7	Data: B node to local B node	BECHO - Send data message to origin
DATABCHQ	Data: B node to CHQ	HQ - Process data message
DATABHQ1	Data: B node to HQ1	HQ - Process data message
DATABHQ2	Data: B node to HQ2	HQ - Process data message
HCOPYCHQ	Hardcopy request: B node to CHQ	CHQHD - Process hardcopy request message and send response to origin
HQ1DGEN	Data: HQ1 to B node Data: HQ1 to HQ2 Data: HQ1 to CHQ	B-ORIGIN - B node stub process HQ - Process data message HQ - Process data message
HQ1GGEN	Graphics request: HQ1 to CHQ	CHQGD - Process graphics request message and send response to origin
HQ1HGEN	Hardcopy request: HQ1 to CHQ	CHQHD - Process hardcopy request message and send response to origin
HQ2DGEN	Data: HQ2 to B node Data: HQ2 to HQ1 Data: HQ2 to CHQ	B-ORIGIN - stub process HQ - Process data message HQ - Process data message
HQ2GGEN	Graphics request: HQ2 to CHQ	CHQGD - Process graphics request message and send response to origin
HQ2HGEN	Hardcopy request: HQ2 to CHQ	CHQHD - Process hardcopy request message and send response to origin

Figure 18. Example 1 Processes

3.3.1.6.1 Processes DATABB01 through DATABB07 The seven Processes DATABB01 through DATABB07 represent the transmitting of data messages from one airbase to another. Each of the Processes consists of a simple call to REQ-I/O which initiates an instance of the Process BECHO (discussed below). BECHO generates and transmits a return message. Each of these seven processes

differs in but one parameter, namely, the node to which the data is transmitted. Below is the graphic representation of DATASBOL:

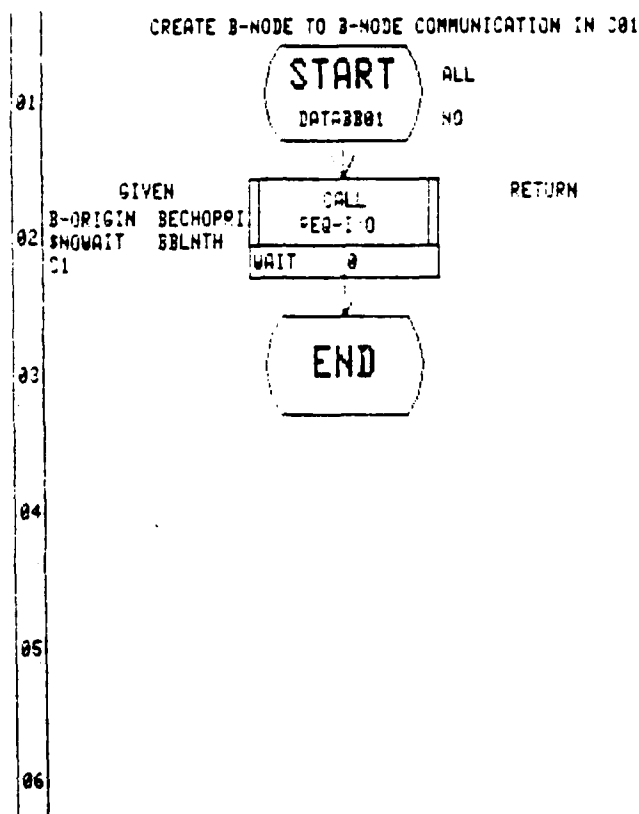


Figure 19. Process DATA8B01

The annotations at the left of the START figure indicate, first, that the Process executes in all nodes--though, in fact, its execution will be confined to "B" nodes. Secondly, the Process has no attached attributes. The parameters to the left of the CALL primitive carry the values accessed by the Message Routing Submodel, of which "REQ-I/O" is the top-level Process.

3.3.1.6.2 Process: BECHO The Process BECHO echos a message back to the node from which it was triggered. It does this by triggering the stub Process B-ORIGIN through the Message Routing Submodel, which is initiated by REQ-I/O. The graphic representation of BECHO is shown in Figure 20.

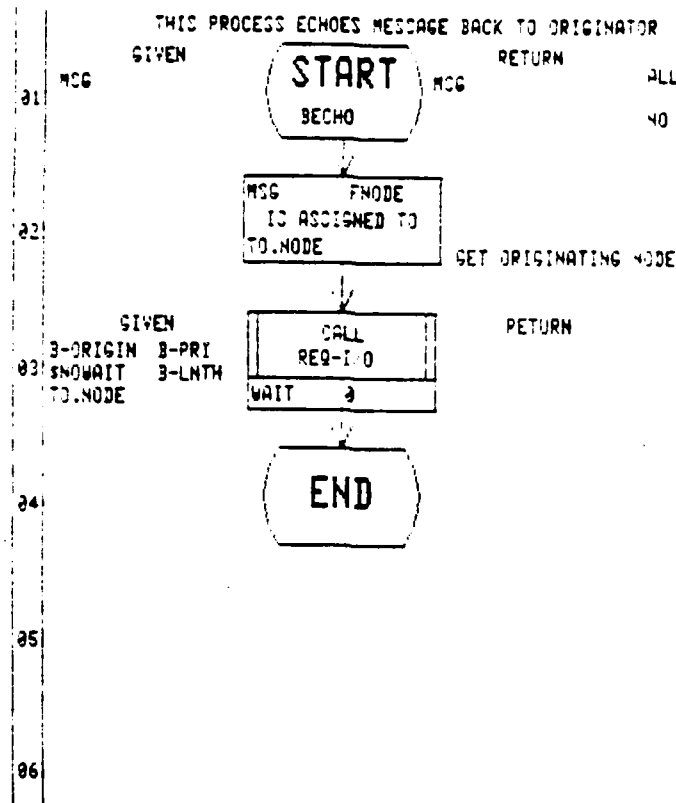


Figure 20. Process BECHO

3.3.1.6.3 Process: B-ORIGIN

The Process B-ORIGIN, which is triggered by HQ1DGEN and HQ2DGEN (discussed below), consists merely of a START symbol and an END symbol. Its purpose is to allow the computation of overhead accrued in the transmission of messages from the two secondary headquarters (HQ1 and HQ2) to the airbases (the B nodes). The graphic picture of B-ORIGIN is shown in Figure 21.

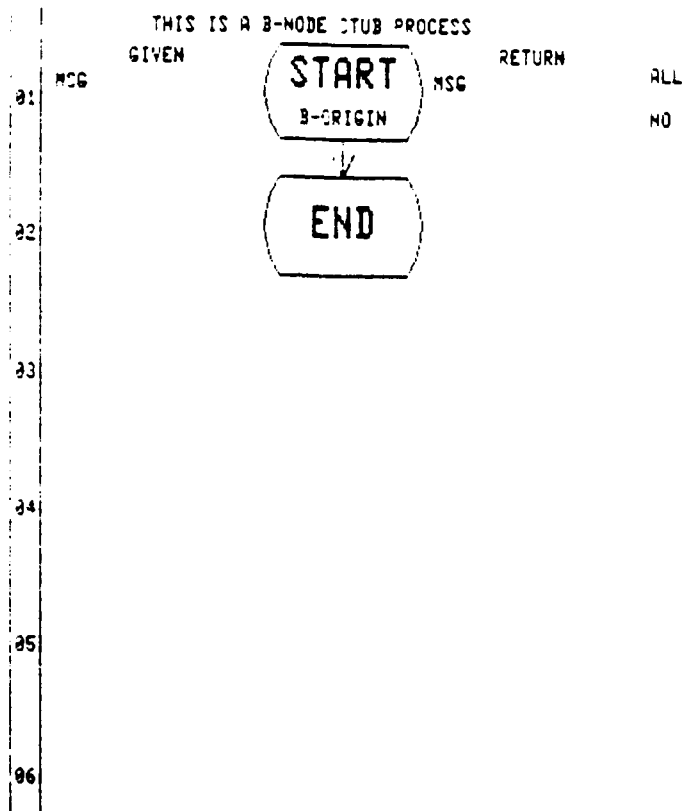


Figure 21. Process B-ORIGIN

3.3.1.7 Processes: DATABCHQ, DATABHQ1 and DATABHQ2 The Processes DATABCHQ, DATABHQ1 and DATABHQ2 all represent the sending of messages from airbases to, respectively, the command headquarters (CHQ) and the two subsidiary headquarters HQ1 and HQ2. Each of these Processes is triggered in all of the B nodes. The effect of these three Processes is to trigger the Process HQ in each of the three destinations, which is done by calling the Process HQ in the CHQ, HQ1 and HQ2. Thus, these three Processes differ only in the destination node that given as a parameter in the CALL to REQ-I/O--the top-level Process in the Message Routing Submodel--of the Process HQ. The Process HQ (discussed below) represents the reception and processing of a message in the destination. The flow-chart representation of DATABCHQ is shown in Figure 22; that of DATABHQ1 in Figure 23 and that of DATADQ in Figure 24.

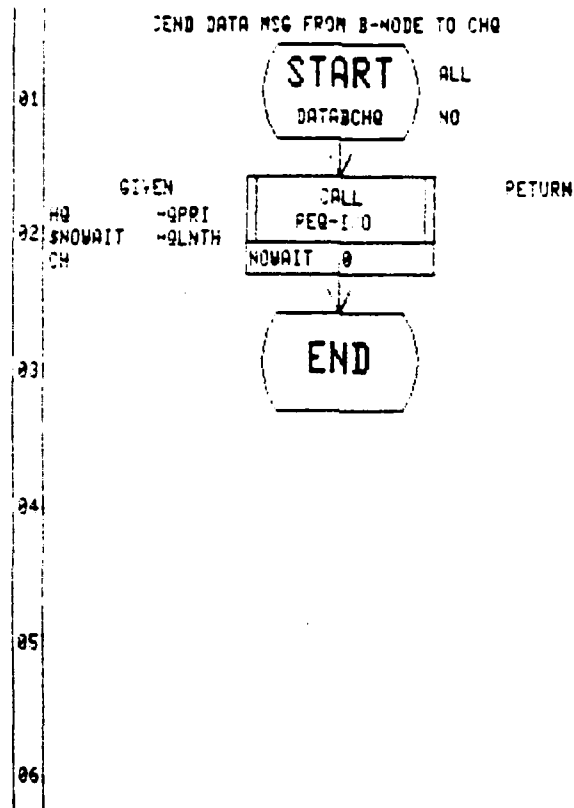


Figure 22. Process DATABCHQ

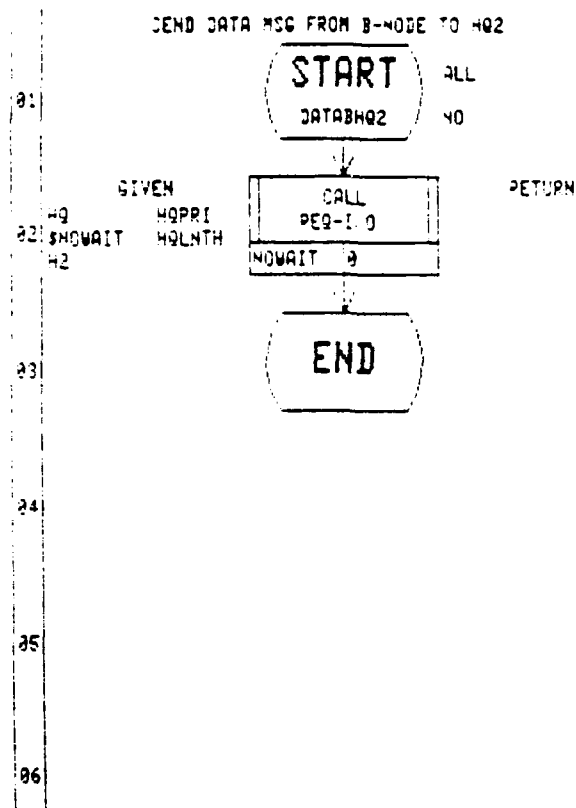


Figure 24. Process DATAHQ2

3.3.1.7.1 Processes: HQ1DGEN and HQ2DGEN The Processes HQ1DGEN and HQ2DGEN represent the generation and transmission of a message at the HQ1 and HQ2 nodes. The generated messages are conceived to be sent to airbases, the command headquarters and the opposite secondary headquarters. Therefore, HQ1DGEN and HQ2DGEN each initiate three Processes: the Process HQ is triggered in the opposite headquarters (HQ1 or HQ2) node and in the CQ node. The Process B-ORIGIN, discussed above, is triggered in B3. These three initiations are effected through calls to REQ-I/O. The graphic version of HQ1DGEN--which is identical to HQ2DGEN except in locus of execution and in the destination of its messages--is given in Figure 25.

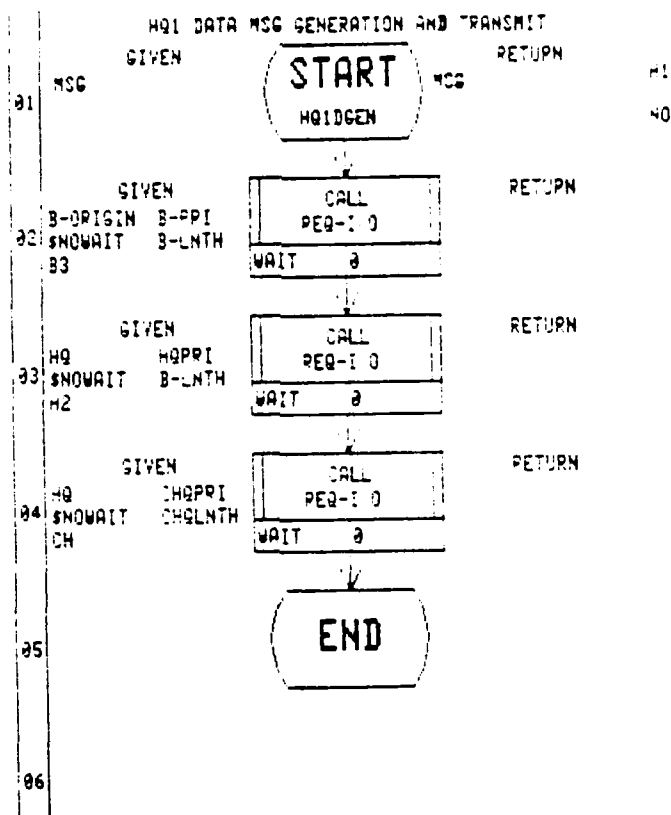


Figure 25. Process HQ1DGEN

3.3.1.7.2 Processes: HQ1GGEN and HQ2GGEN The Processes HQ1GGEN and HQ2GGEN represent the transmission of a graphics request from each of the two secondary headquarters to the command headquarters. These Processes both have the effect of initiating a Process in the CHQ node, namely CHQGD, to represent the reception and processing of the graphics request. HQ1GGEN is represented in flow-chart form in Figure 26. HQ1GGEN and HQ2GGEN are identical except for the the nodes in which they execute.

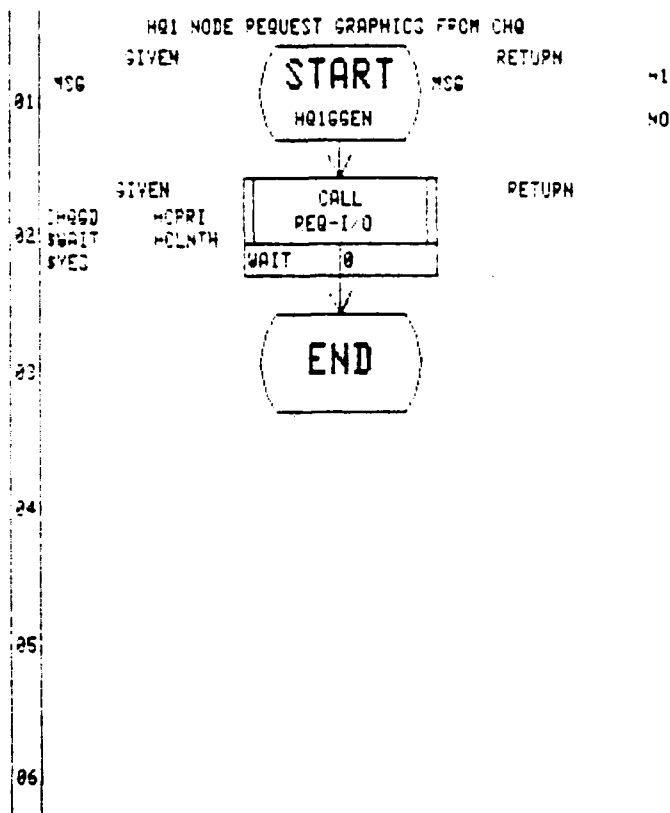


Figure 26. Process HQ1GEN

3.3.1.3 Processes: HQ1HGEN and HQ2HGEN

The Processes HQ1HGEN and HQ2HGEN represent the generation of a request for nardcopy from the secondary headquarters to the command headquarters. They are triggered in the nodes HQ1 and HQ2 and in turn call a Process CHQHD in the CHQ node. HQ1HGEN is identical to HQ2HGEN except in node of execution.

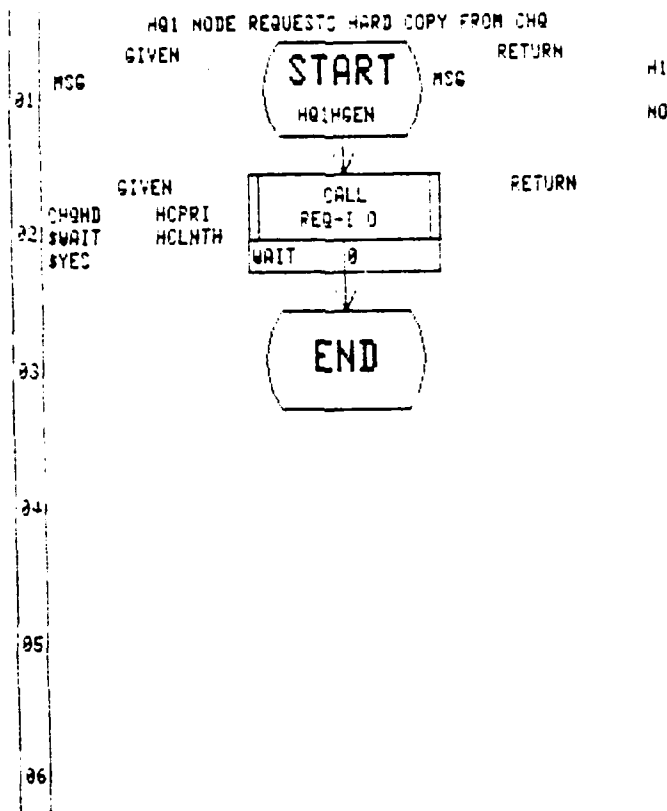


Figure 27. Process HQLHGEN

3.3.1.9 Process: CHQGD The Process CHQGD is much like CHQHD. It represents the processing of a graphics request from the two secondary headquarters. It is called by the Processes HQLGGEN and HQL2GGEN. In it, a reference to the Item MSG is passed from the calling Process. An attribute of MSG is accessed by the ASSIGN primitive and its value is assigned to a local variable. That value is then used to evaluate (with the EVAL primitive) the overhead time entailed in the response to the request. An ACTION primitive then takes up the calculated amount of time and the reference to MSG is returned to the calling Process. The flow-chart rendering of CHQGD is shown in Figure 28.

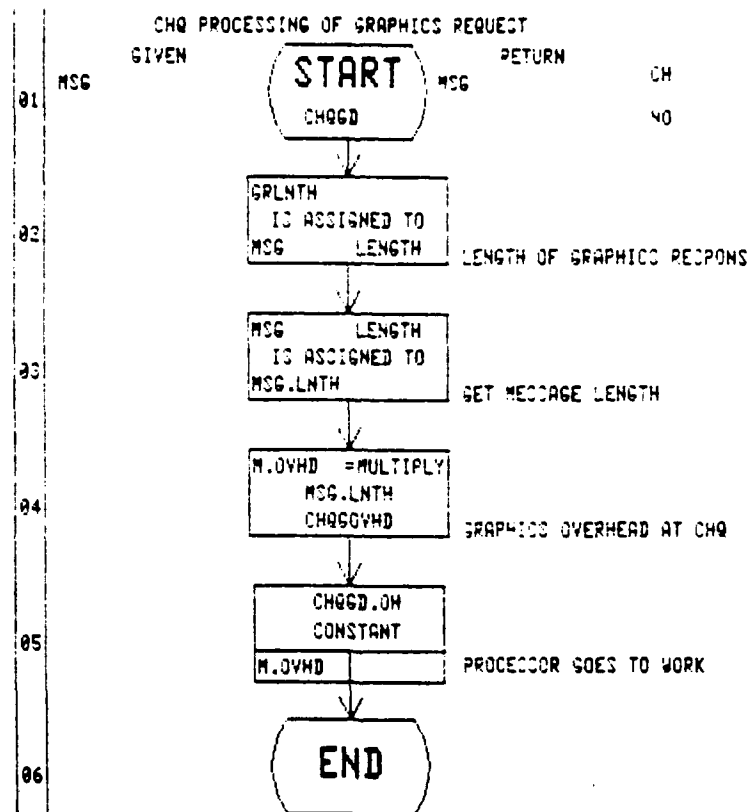


Figure 28. Process CHQGD

3.3.2 Load: LOADS01 Load LOADS01 describes the Load placed on the communication system from hosts connected to S01. Remembering the assumption made that all hosts connected to a switch node can be represented by one node with the channel transfer constraint relaxed, LOADS01 specifies the Processes which initiate at node B01.

There are two types of messages which originate at B1, data request messages with a mean message length of 750 characters, and nardcopy display request messages with a mean message length of 200 characters. Display request messages from node B01 request data be sent to node H01, the CHQ node and echoed back to B1.

In order to specify the Load to AISIM, the messages rates given in Figures 7 and 8 in msg/sec, poisson distributed need to be expressed by an interarrival rate given in sec/msg. The exponential distribution represents a poisson arrival pattern for interarrival times.

The computation of mean interarrival rate for message generation at node B01 is computed in the following way. Data messages have a message rate of .517 msg/sec (Figure 7 -- Message traffic

Characteristics). This is distributed 94% to all S nodes (Figure 7 -- Message Traffic Sources). That is, the message rate for the system originating at S nodes is .517 msg/sec x .94 = .486 msg/sec. Distributing this to seven nodes equally means that any one S node must have a message generation rate of $(.486 \text{ msg/sec})/7$ S-nodes = .07 msg/sec per S node must. Inverting this for interarrival time gives 14.4039 secs/msg per S node must. This is the interarrival rate for data request Process triggering which applies to Processes DATA801, DATA8H1, and DATA8H2 which originate at source node B01 (Figure 3). The computation for Hardcopy display request generation from node B01 uses a similar method to compute the mean interarrival time for HCOPI8H2 Processes originating at B01. The time for this is 14.643 sec/msg.

The definition of LOADS01 is shown in the following figure. Loads originating at other nodes B02 through B07, H01 and H02, are calculated in a similar manner from the data in Figures 7 and 8.

LOAD: LOADS01						
DESCR: S1 LOAD ORIGINATING AT BASE B01						
<u>NODE1</u>	<u>NODE2</u>	<u>NODE3</u>	<u>NODE4</u>			
B01						
<u>NODE5</u>	<u>NODE6</u>	<u>NODE7</u>	<u>NODE8</u>			
<u>PROCESS</u>	<u>MAX #</u>	<u>SCH MTD</u>	<u>MEAN</u>	<u>DELTA</u>	<u>PRIORITY</u>	
DATA801	1000	EXPONENT	14403		0	
DATA8H1	1000	EXPONENT	14403		0	
DATA8H2	1000	EXPONENT	14403		0	
HCOPI8H2	1000	EXPONENT	14643		0	

Figure 29. Example 1 Load Originating at Node S01

3.4 Analysis of Results

3.4.1 Performance Measures Each node and channel on the AISIM architecture represents a resource in the system. The statistics for the performance of the resources described in section 3.1.1 are found in the AISIM Resource Report.

3.4.1.1 Communications Queue Time Each channel is represented by one or two resources depending on whether the channel is half or full duplex. Communications queue time is expressed as the RESOURCE WAIT TIME. The MEAN, STD DEV, MINIMUM and MAXIMUM figures refer to the mean wait time, standard deviation about the mean, minimum wait time and maximum wait time for messages. These figures are based on the TOTAL NUMBER which is the number of samples collected for the statistic. Each sample represents the wait time of a message for the channel. These values are given for each channel.

A "collective" statistic for communication queue time can be found in the Process Report under RESOURCE WAIT for the Process CHLIO. This heading lists the SUM, MEAN, STANDARD DEVIATION, MINIMUM and MAXIMUM wait times accumulated for the Process waiting on the ALLOC Primitive. Each instance of CHLIO corresponds to a separate instance of a message. Since there is only one ALLOC Primitive, the RESOURCE WAIT statistic collects the time for all messages in CHLIO.

3.4.1.2 Communication Channel Queue Length The channel queue length statistics are found in the Resource Report for each channel in the architecture. The statistic is listed under the # WAITING heading, which is the collection of samples taken by the AISIM simulator each time the Resource wait queue is changed. The # WAITING statistics are time weighted, which means that the MEAN # WAITING statistic takes into account how long the Resource wait queue was a given length.

3.4.1.3 Processor Queue Time The queue time for processors is shown in the AISIM Resource Report for each processor node identified on the system architecture. As for channels, processor queue time is listed as the Resource Wait Time in the report.

3.4.1.4 Processor Queue Length The queue length for processors is shown in the AISIM Resource Report for each processor node identified on the system architecture. As for channels, processor queue length is listed as the Resource # WAITING in the report.

3.4.1.5 Communications Channel Utilization The communication channel utilization for each channel in the architecture is listed in the Resource Report under the # BUSY heading. The MEAN # BUSY statistic gives the percent utilization of the channel. The MEAN # BUSY is a time weighted statistic which is updated by the AISIM simulator each time a channel is made idle by a DEALLOC Primitive. The time the channel was busy is added to the total busy time. At the end of the simulation this is divided by the total simulation clock time.

3.4.1.6 Processor Utilization The processor utilization for each processor in the architecture is listed in the Resource Report under the # BUSY heading. The MEAN # BUSY statistic gives the percent utilization of the channel. This statistic is time weighted, which is updated by the AISIM simulator each time a processor is made idle by a DEALLOC Primitive. The time the processor was busy is added to the total busy time. At the end of the simulation this is divided by the total simulation clock time.

3.4.1.7 Other Performance Measures Based on the way the model has been built, it is not possible to get message response times by source and destination nodes. For those processes which call REQ-I/O with the WAIT option, message response time can be

obtained by function from the output listing under the Process Report. This appears under the TOTAL heading. This heading lists the TOTAL SAMPLES, SUM, MEAN, STANDARD DEVIATION, MINIMUM and MAXIMUM completion times of the Process. For Processes HARD-COPY, HCPYCHQ, HQLGGEN, HQLHGEN, HQ2GGEN and HQ2HGEN, the response time is accumulated under this statistic.

3.4.2 Validating the Model The node and channel utilizations for all resources in the architecture can be calculated analytically. Utilization is a function of the Load (message generation rate) and the processing times. For channels, the processing time is dependent on message lengths and channels transmission speeds. For processors, the processing time is dependent on message lengths and processor cycle speeds. These figures are available and can be calculated. Comparing expected utilization of resources to simulation generated results provides a first level of validity.

Node Utilizations (in %)

<u>Node</u>	<u>Predicted</u>	<u>AISIM</u>
		<u>Simulation Results</u>
S1	5.22	4.8
S2	8.77	8.8
S3	9.77	9.9
S4	21.23	20.8
S5	4.80	5.0
S6	9.97	3.3
S7	36.37	37.1
H1	1.91	1.7
H2	1.91	1.3
CH	56.60	69.8

Figure 30. Example 1 Node Utilization Results

Channel Utilization (in %)

<u>Channel</u>	<u>Connecting</u>	<u>Predicted</u>	<u>Simulation Results</u>
S2S1.B	S1 to S2	2.74	2.7
S3S1.B	S1 to S3	4.34	4.7
S2S1.A	S2 to S1	20.09	19.8
S2S7.A	S2 to S7	5.48	5.9
S4S3.B	S3 to S4	4.15	4.4
S3H1.A	S3 to H1	12.53	12.6
S4S3.A	S4 to S3	38.46	40.1
S6S4.B	S4 to S6	38.46	35.6
S4S7.A	S4 to S7	5.05	4.7
S5S6.A	S5 to S6	2.17	2.5
S5S7.A	S5 to S7	2.74	2.7
S6S4.A	S6 to S4	4.15	3.5
H2S6.B	S6 to H2	12.53	11.3
S2S7.B	S7 to S2	35.85	37.6
S4S7.B	S7 to S4	43.97	44.8
S5S7.B	S7 to S5	17.93	20.7
CHS7.B	S7 to S3	1.75	1.8
S3H1.B	H1 to S3	.89	1.0
H2S6.A	H2 to S6	.89	.7
CdS7.A	Cd to S7	13.69	14.3

Figure 31. Example 1 Channel Utilization Results

3.5 Conclusions

The AISIM model of the communication network is very representative of the system description. The approach taken for this model demonstrates a technique which can be used very successfully in simulation modelling. That is, building a general sub-model to represent a specific function in the system. The advantage of eliminating dependence of model components is that the fewer dependencies in a model, the more easily adapted it is to other applications.

Example 1 is certainly not a simple problem. For that reason the solution approach shown is not trivial. Still, the model implementation plan consists of a manageable set of discrete steps towards model implementation.

4. Example 2 - Loop Communication Network

Example 2 demonstrates the application of AISIM to another communication system utilizing a loop protocol. Unlike example 1 where low level protocols of the subnet node communication was not described, modelling a loop involves addressing some basic elements of the communications, like buffers and slots.

A loop is described most basically as single direction, circular communication. This example concentrates on a loop system using a Pierce protocol.

This example shows how it is possible to use parts of previously developed models to quickly generate a new model of a totally different system.

4.1 Input

4.1.1 Mission Concept Nodes in a network communicate through a Pierce loop architecture. A Pierce loop is described by a circular linked architecture. Messages communicate between nodes by circulating around the architecture via message slots. The nodes are the "users" of the communication loop. Each node represents a processor servicing its own class of users. Processing and data is distributed to the nodes, necessitating the inter-node communication.

4.1.2 Problem Perspective Several messages may be in the network simultaneously. The performance of the communication system using a varying mix of messages will indicate whether messages are efficiently handled. The following are definitions of performance criteria for this example.

1. Queueing time - Time elapsed from message generation until start of placement on the loop by the transmitter.
2. Transmission time - Time elapsed from the start of message placement on the loop until it is received at its destination and removed from the loop.
3. Total Message Transmission Time - Sum of queueing and transmission times.

4.1.3 System Description The following characteristics apply to the network:

1. The network consists of six nodes.
2. Messages are generated randomly at each node and uniformly addressed to the other five nodes.
3. A message consists of 1 packet. Each packet consists of 36 characters.

4. In the Pierce loop, fixed length slots "circulate" around the loop. A lead field of the slot indicates whether or not the following frame is occupied. In the absence of a message, a host may insert a message (or portion thereof) into the available slot.

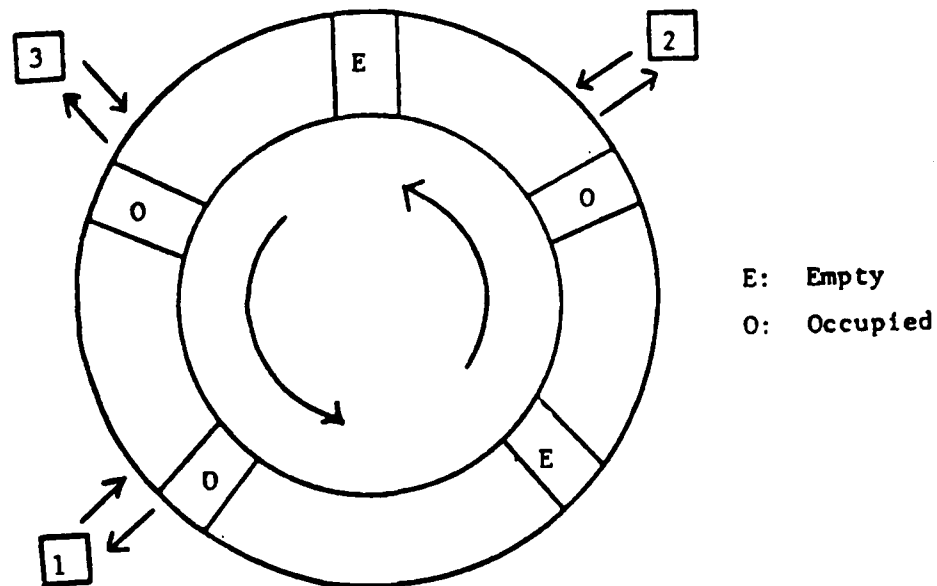


Figure 32. Pierce Loop Diagram

4.2 Preliminary Analysis

AISIM is applicable to this problem. Message communication in a loop network can be described by discrete events. The discrete events of interest in the loop communication are the following:

1. Message created at a node.
2. Message queued at origin for slot.
3. Start slot transmission on loop.
4. Slot received at loop interface unit.
5. Slot routed to next loop interface unit.
6. Message received at destination, processed and eliminated.

Each of these events can be modeled by procedural operations.

Many slots circulate on the loop simultaneously. Slots are processed in parallel at nodes.

Resources in the system consist of node processors, node storage buffers, and channel connectors between nodes.

The incidence of messages introduced into the system describes the external loading.

A loop is a subset of an interconnected network architecture. Each node has exactly 2 connectors, each to adjacent nodes in the architecture. Communication is implemented in only one direction.

4.2.1 Problem Definition The elements to be analyzed in the loop communication system can be grouped into two categories. The first addresses unidirectional message communication from any source node to any destination node around the loop through the loop interfaces. Here the model will represent random occurrences of messages in the system at source nodes, to be sent arbitrarily to nodes in the loop. This high level approach will generate throughput statistics on messages based on the inter-arrival rates of messages. Processing and queueing for processors and channels will be represented.

The lower elements of this system -- slot synchronization, slot processing, and message buffering will be represented in the extended model.

Although basic to loop communications, varying message lengths will not be addressed. It is assumed that the loads representing the external environment of the system take into account the software which assembles and disassembles messages.

4.2.2 Definition of Objective

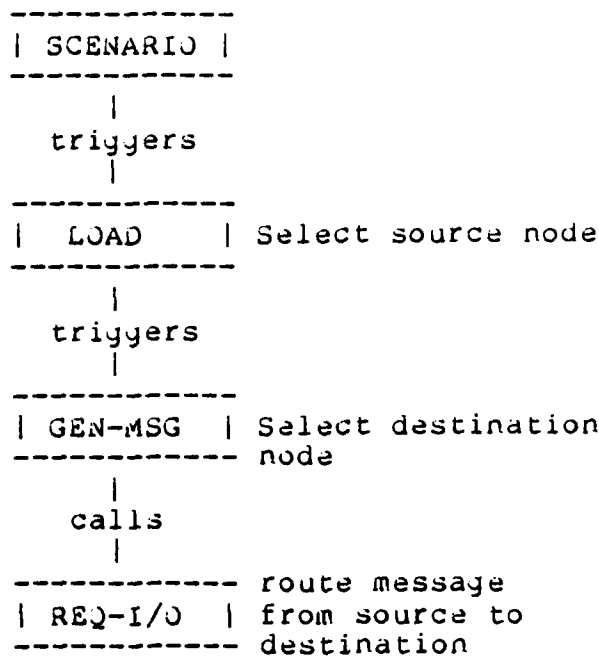
The objective of modelling the loop communication network is to produce quantifiable results for the performance measures in the system description. The model will be built first to satisfy the high level elements of the system described in the problem definition (unidirectional loop architecture, random message load). The model will be extended to include the lower level elements (slots and message buffering).

4.3 Model Build

4.3.1 Design, Plan and Construction of the Model

4.3.1.1 Model Design A two-step approach is used to design the model based on the two categories of modeling problems identified in the problem definition. The preliminary design makes use of the message routing submodel described in Example 1. The choice of this submodel at this time is based on experience that it is

easier to start from some known model than it is to start from scratch if it looks like the elements addressed are similar enough. Unidirectional flow and random origin and destination nodes fit the message routing submodel assumptions.



4.3.1.2 Model Implementation Plan The sequence for constructing the model of the Pierce loop communication network is described below:

1. Define Pierce loop architecture.
2. Interface message routing submodel (from Example 1) to loop model.
3. Build generation Process(es).
4. Verify message routing submodel on loop architecture.
5. Define system parameters and full loading.
6. Analyze high level loop model.
7. Design lower level elements into model (slots and buffers).
8. Build models of lower level elements.
9. Verify new network model.
10. Analyze results.

4.3.2 Define Pierce Loop Architecture Defining the Pierce loop architecture for a six host network is a straightforward task. Each host and loop interface unit can be represented by a node. This assumes that loop interface units are embedded in hosts and not a separate hardware element which needs to be addressed as a network element. This is reasonable based on the information in the system description. Each node has two links representing communications channels to adjacent nodes. The legal path table for this network describes the directional flow of messages.

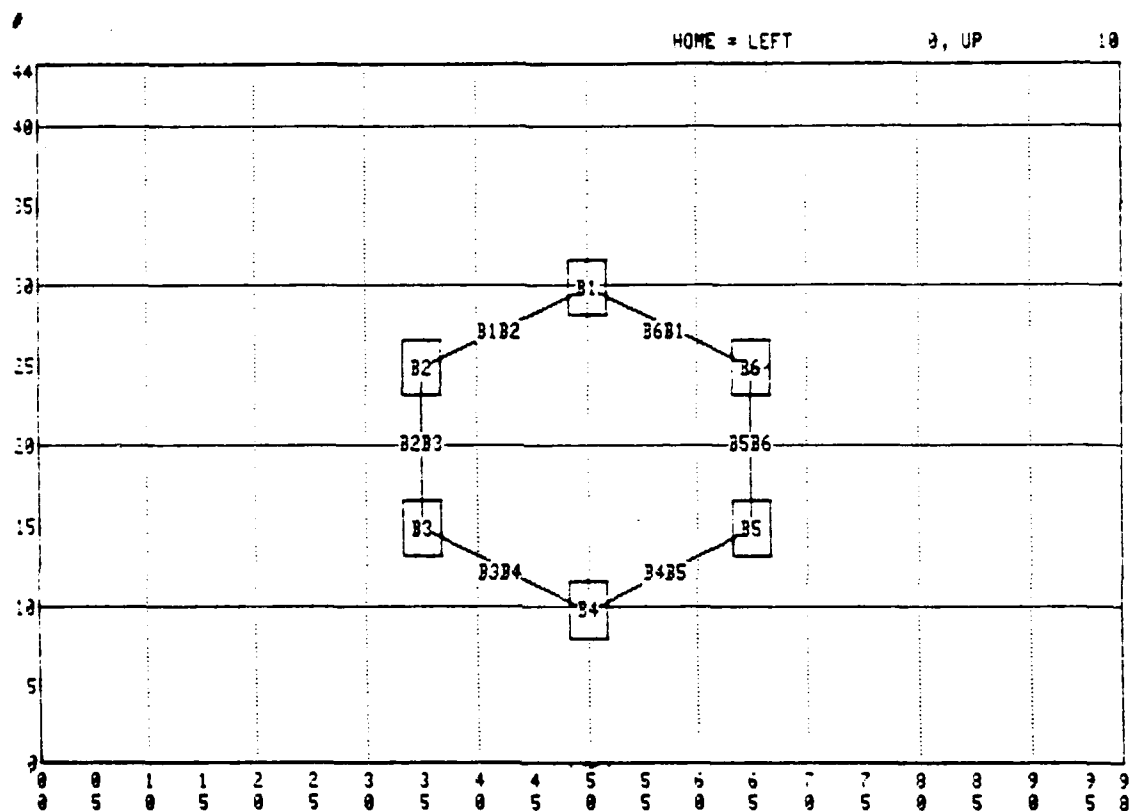


Figure 33. Example 2 Loop Architecture

FROM	TO	NEXT	LINK	FROM	TO	NEXT	LINK
B1	B2	B2	B1B2	B4	B1	B5	B4B5
B1	B3	B2	B1B2	B4	B2	B5	B4B5
B1	B4	B2	B1B2	B4	B3	B5	B4B5
B1	B5	B2	B1B2	B4	B5	B5	B4B5
B1	B6	B2	B1B2	B4	B6	B5	B4B5
B2	B1	B3	B2B3	B5	B1	B6	B5B6
B2	B3	B3	B2B3	B5	B2	B6	B5B6
B2	B4	B3	B2B3	B5	B3	B6	B5B6
B2	B5	B3	B2B3	B5	B4	B6	B5B6
B2	B6	B3	B2B3	B5	B6	B6	B5B6
B3	B1	B4	B3B4	B6	B1	B1	B6B1
B3	B2	B4	B3B4	B6	B2	B1	B6B1
B3	B4	B4	B3B4	B6	B3	B1	B6B1
B3	B5	B4	B3B4	B6	B4	B1	B6B1
B3	B6	B4	B3B4	B6	B5	B1	B6B1

Figure 34. Example 2 Legal Path Table

4.3.3 Interfacing Message Routing Submode to Loop

Model Including the message routing submodel in the loop model is done by merging the entities developed in Example 1 for message routing, the Item MSG, the six Processes, the Actions, and global variables, into a loop model. This is done with the AISIM Library User Interface. The message routing submodel is interfaced to the loop model by defining the Resources to have the attributes addressed in the Processes and defining the Processes which invoke REQ-I/O.

4.3.4 Process: GEN-MSG A Process which controls the generation of messages determines source and destination nodes for each message and indicates a Process to be initiated at the destination node to handle the message. Processes can be oriented in a network through the NODE fields of a Load entity. This is a natural mechanism for determining source nodes for messages. The source node is the node which executes the message generation Process.

There are many strategies which can be considered for the logic to determine destination nodes for messages generated at a source node. For economy in effort, a good approach would be to define a single Process which would execute in all nodes and distribute messages to all other nodes uniformly. This can be done by defining an AISIM Table, indexed by a random integer, which evaluates to the name of a node in the Architecture. The definition of this Table is shown below:

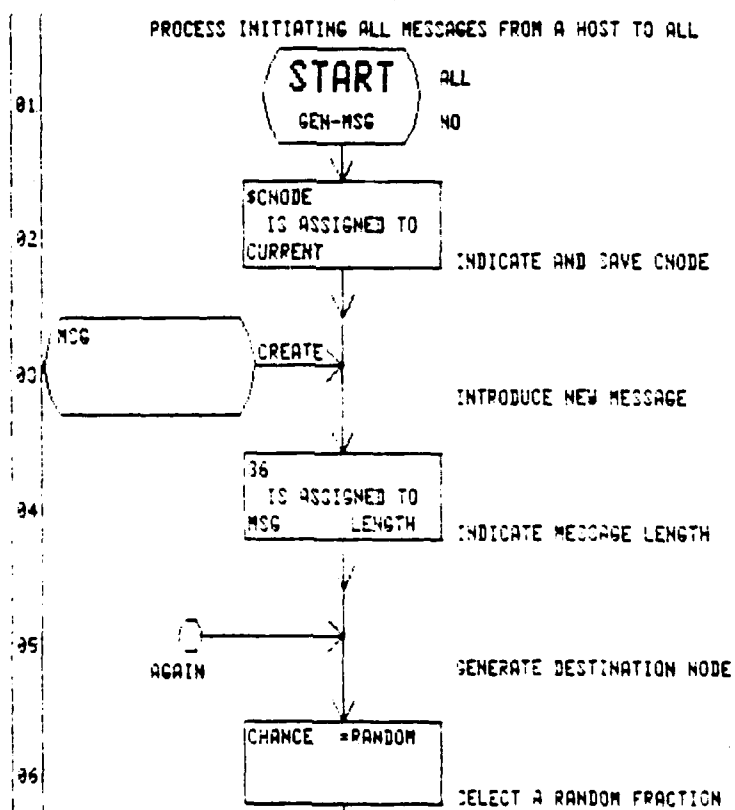
TABLE NAME: NODE-TBL TYPE: A
DESCRIPTION: TABLE WITH NAME OF EACH NODE TO BE ACCESSED
 RANDOMLY

<u>X-VALUE</u>	<u>Y-VALUE</u>
0	B1
1	B2
2	B3
3	B4
4	B5
5	B6

Figure 35. Example 2 NODE-TBL

To use this Table, a Process selects a random fraction, scales the fraction to the number of entries in the Table by multiplying and truncates the product to match an X-VALUE in the Table. These three transformations are done using the EVAL Primitive. Since the message generation Process can execute in all nodes, it is necessary to check if the randomly selected node is the source node. If so, another node must be generated. This technique for generating uniformly distributed selections from a range of values is standard practice in simulation.

The message generation Process, GEN-MSG, triggers the initial Process in the message routing submodel, REQ-I/O, with the parameters to generate a message and route it to a destination.



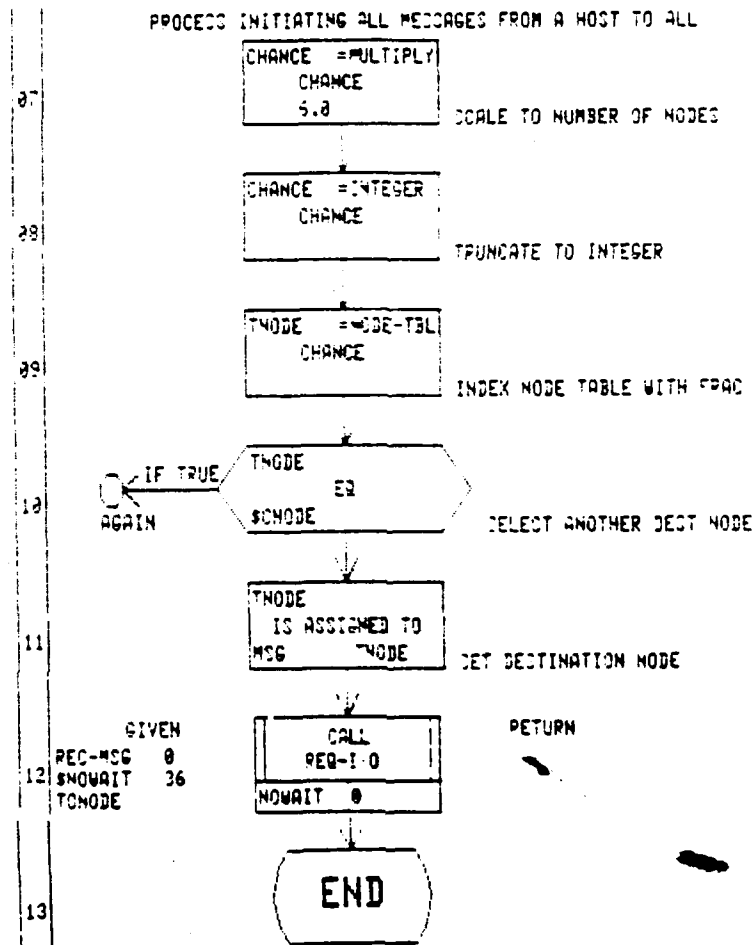


Figure 36. Process GEN-MSG

4.3.5 Verifying the Message Routing Submodel and Loop Model

4.3.5.1 Single Thread Scenario and Load A single thread Scenario, TEST1, initiates one instance of GEN-MSG at node 31. A trace is generated to follow the sequence of transactions for the message to determine if the message is properly routed to its destination.

SCENARIO NAME: TEST1 PERIOD LENGTH: 60
 DESCRIPTION: SINGLE THREAD TEST CASE FOR PIERCE LOOP
 PERIODS

1

TRIGGER	SCH	TIME	PRIORITY
LOAD1	0		0
TRACE	0		0

LOAD NAME: LOAD1
 DESCRIPTION: GENERATE 1 MESSAGE AT NODE 1 AFTER 1 SECOND
 NODE
 B1

PROCESS	MAX #	SCH MTD	MEAN	DELTA	PRIORITY
GEN-MSG	1	INTERVAL	1	0	0

Figure 37. Example 2 Single Thread Scenario and Load

4.3.5.2 Single Thread Trace The single thread trace of Process execution shows the sequence of Processes routing one message from node B1 to its destination. The Process, REC-MSG, is executed at the destination. The trace verifies that the correct sequence of Processes occurs.

T=	0.	N = NONE	P = TRACE	TRACE: ON
T=	0.	N = NONE	P= TRACE	END
T=	1.00000	N = B1	P= GEN-MSG	START
T=	1.00000	N = B1	P= GEN-MSG	CALL REQ-I/O
T=	1.00000	N = B1	P= GEN-MSG	END
T=	1.00000	N = B1	P= REQ-I/O	START
T=	1.00000	N = B1	P= REQ-I/O	CALL ESR-CALL
T=	1.00000	N = B1	P= ESR-CALL	START
T=	1.00000	N = B1	P= ESR-CALL	CALL ROUTER
T=	1.00000	N = B1	P= ROUTER	START
T=	3.00000	N = B1	P= ROUTER	CALL CHLID
T=	3.00000	N = B1	P= ROUTER	END
T=	3.00000	N = B1	P= CHLID	START
T=	3.00000	N = B1	P= CHLID	ALLOCATE B1B2
T=	3.00000	N = B1	P= ESR-CALL	END
T=	3.00000	N = B1	P= REQ-I/O	END

T=	31.00000	N = B2	P=	CHLIO	DEALLOC	B1B2
T=	31.00000	N = B2	P=	CHLIO	CALL	IHANDLER
T=	31.00000	N = B2	P=	CHLIO	END	
T=	31.00000	N = B2	P=	IHANDLER	START	
T=	31.00000	N = B2	P=	IHANDLER	ALLOCATE	B2
T=	33.00000	N = B2	P=	IHANDLER	DEALLOC	B2
T=	33.00000	N = B2	P=	IHANDLER	CALL	CHLIO
T=	33.00000	N = B2	P=	IHANDLER	END	
T=	33.00000	N = B2	P=	CHLIO	START	
T=	33.00000	N = B2	P=	CHLIO	ALLOCATE	B2B3
T=	61.00000	N = B3	P=	CHLIO	DEALLOC	B2B3
T=	61.00000	N = B3	P=	CHLIO	CALL	IHANDLER
T=	61.00000	N = B3	P=	CHLIO	END	
T=	61.00000	N = B3	P=	IHANDLER	START	
T=	61.00000	N = B3	P=	IHANDLER	CALL	CONTROL
T=	61.00000	N = B3	P=	IHANDLER	END	
T=	61.00000	N = B3	P=	CONTROL	START	
T=	61.00000	N = B3	P=	CONTROL	ALLOCATE	B3
T=	63.00000	N = B3	P=	CONTROL	CALL	REC-MSG
T=	63.00000	N = B3	P=	REC-MSG	START	
T=	63.00000	N = B3	P=	REC-MSG	END	
T=	63.00000	N = B3	P=	CONTROL	DEALLOC	B3
T=	63.00000	N = B3	P=	CONTROL	END	

Figure 38. Example 2 Single Thread Trace

4.3.5.3 Multiple Thread Scenario and Loads A reasonable multiple thread Scenario for verifying the model initiates six messages, simultaneously, one at each host node. As the messages circulate around the loop, it is possible to trace the sequence of events and detect queueing. The simulation results produced should be verifiable by the trace output. The selection of this Scenario is based on the objective of defining a small experiment which can be hand verified if necessary in order to prove the model is executing correctly.

SCENARIO: TEST2 PERIOD LENGTH: 60
DESCRIPTION: MULTIPLE THREAD SCENARIO TEST
PERIODS: 1

<u>CALLS:</u>	<u>TRIGGER</u>	<u>SCH TIME</u>	<u>PRIORITY</u>
STARTB1	0	0	
STARTB2	0	0	
STARTB3	0	0	
STARTB4	0	0	
STARTB5	0	0	
STARTB6	0	0	
TRACE	0	0	

LOAD: STARTB1
DESCR: SINGLE OCCURRENCE OF GEN-MSG AT B1

<u>NODE1</u>	<u>NODE2</u>	<u>NODE3</u>	<u>NODE4</u>
B1			

<u>NODE5</u>	<u>NODE6</u>	<u>NODE7</u>	<u>NODE8</u>

<u>PROCESS</u>	<u>MAX #</u>	<u>SCH MTD</u>	<u>MEAN</u>	<u>DELTA</u>	<u>PRIORITY</u>
GEN-MSG	1	START			0

Figure 39. Example 2 Multiple Thread Test Case

Loads STARTB2, STARTB3, STARTB4, STARTB5, and STARTB6 are similar to STARTB1 except in the name in the NODE1 field.

4.3.6 Full Loading Scenario Simulation runs of the Pierce loop network are required to generate response data for varying mean inter-arrival times of messages at nodes. A full loading Scenario specifies for each node a load which initiates the generation of messages, distributed over time by an exponential mean inter-arrival rate. This models Poisson arrivals, i.e. random message generation.

An example of a full loading Scenario, generating messages exponentially distributed with a mean inter-arrival time of 240 seconds at each node is shown in the accompanying figure.

SCENARIO: RUN240 PERIOD LENGTH: 240
DESCRIPTION: SIX NODE PIERCE LOOP WITH MESSAGES EVERY 240
SECS.

PERIODS: 1 2 3 4 5 6 7
 8 9 10

<u>CALLS</u>	<u>TRIGGER</u>	<u>SCH TIME</u>	<u>PRIORITY</u>
LOADB1		0	0
LOADB2		0	0
LOADB3		0	0
LOADB4		0	0
LOADB5		0	0
LOADB6		0	0

LOAD: LOADB1
DESCR: MESSAGES AT SOURCE B1 EXPONENTIALLY DISTRIBUTED BY
C.ARRIVE

<u>PROCESS</u>	<u>MAX #</u>	<u>SCH MTD</u>	<u>MEAN</u>	<u>DELTA</u>	<u>PRIORITY</u>
GEN-MSG	10000000	EXPONENT	C.ARRIVE		0

CONSTANT: C.ARRIVE

VALUE: 240

DESCR: MEAN INTER-ARRIVAL RATE OF MESSAGES IN SECONDS

Figure 40. Example 2 Full Loading Scenario and Entity Definitions

Loads LOADB2 through LOADB6 are defined similar to LOADB1. The Constant C.ARRIVE is used to specify the inter-arrival time in order to make changing this value easier.

4.3.7 Extending the Model to Include Slots and Buffers The model described so far represents the Pierce loop architecture and functional processing. It is easily built. Using AISIM, the Pierce loop model can be built and running with just a few hours of work. This is made possible by making use of the message routing model developed in Example 1, which performs much of the logic for the loop model. As yet, the model does not include one-to-one representations of some of the lower level elements in the Pierce loop communication. These include the message slots, node buffers and slot synchronizing which are important parts of this communication protocol. To include these elements in the model, the model built so far can be modified.

The first problem with modelling slots and buffers is to decide what AISIM entities have dynamic characteristics which map to these elements. This is not as straightforward as with the loop

interface units and channels which clearly can be identified as Resources.

Slots are a type of resource in that a slot is either occupied or empty. Messages wait for empty slots. But slots also have a physical location in the network which changes over time. They can be viewed as transient elements which are normally represented in AISIM by Items. The circulation of slots is the most fundamental characteristic of this element so it is best to consider slots as Items and handle utilization of the slots (normally associated with AISIM Resources) with some other mechanism.

There is a similar problem with modelling buffers. Buffers are implemented using a Resource which is storage. Buffers are ordered holding areas for messages, though, so they can be viewed as AISIM user-defined queues. The second view is the more natural one so it is selected.

The two decisions for modelling slots and buffers have the following implications on the model of the Pierce loop.

1. Slots circulate around the loop. Slots contain an attribute which identifies the current node a slot is at. Initially slots must be given a current node and started circulating.
2. Slots contain messages when occupied.
3. When slots are empty, a message can be removed from a host buffer and placed in the slot.
4. Slots are received at nodes and forwarded to the next node on the loop.
5. All messages generated at a host are placed in the host's buffer.

From this use of functions of the Pierce loop system, it is possible to identify four AISIM Processes to define.

STARTUP - This Process is executed once in every node at the start of the simulation. STARTUP models creating a slot and initiating its circulation around the architecture.

GEN-MSG - This Process executes in every node and models generation of a message, determining the destination node for the message, and filing the message in the buffer associated with the node.

FORWARD - This Process models moving a slot from one node to the next node in the loop and interrupting the receiving node.

REC-SLOT - This Process models the logic of the node loop interface unit when a slot is received. If the slot is occupied and the message is destined for the receiving node, the message is processed. If the message is for another node, the slot is forwarded. If the slot is empty, a message is placed in the slot if one is waiting and the slot is forwarded.

The Process FORWARD performs the logic done by the message routing Processes. It would be possible to modify the message routing Processes to route slots and this would be one approach. Another approach which could be considered is to redo the message routing making use of some characteristics of loop architectures. This second approach is shown for comparison purposes.

The structure of the loop model using slots and buffers is shown.

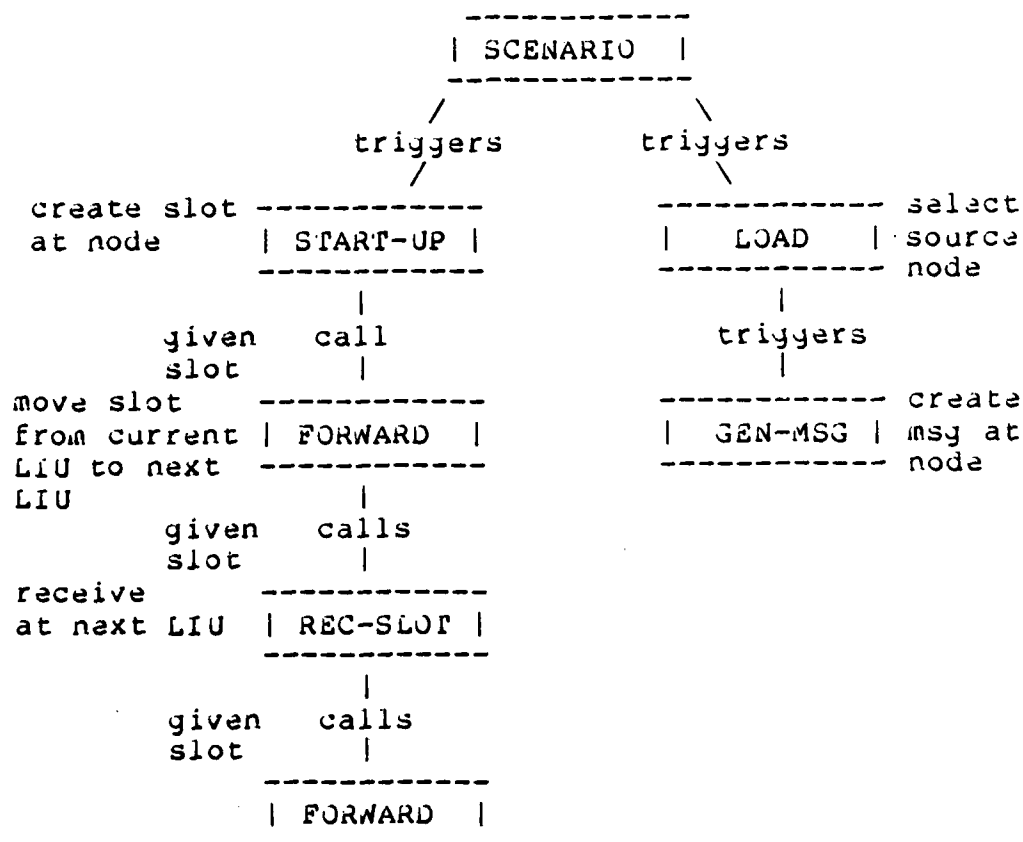


Figure 41. Example 2 Structure With SLOTS And BUFFERS

4.3.7.1 Entity Attributes To support the flow of the Processes, it is necessary to define attributes for the slots and loop interface units. The attributes are referenced in the Process. Each slot item has three attributes described below.

<u>ENTITY TYPE</u>	<u>ENTITY NAME</u>	<u>ATTR NAME</u>	<u>ATTR VALUE</u>	<u>DESCRIPTION</u>
ITEM	SLOT			Item representing slot.
		CNODE	\$CNODE	The node name where each slot is located at any time.
		LENGTH	36	The size in characters of each slot.
		MSG	0 or MSG	If the slot is occupied, this attribute contains an Item, MSG. If the slot is empty, this attribute is 0.

An alternate mechanism for routing slots around the loop utilizes the NEXT attribute defined for loop interface units, nodes B1 through B6. This approach simplifies the model by taking advantage of the uni-directional flow of slots. This approach means that the Architecture and Processes can not be made independent. Changes to the Architecture to model more or less nodes on the loop will need to be accompanied by the definition of loop interface unit node attributes. In the example below, the attribute definitions for node B1 are shown. BUFFER1 is the Queue associated with B1 for holding messages prior to putting them in slots on the loop. B2 in the NEXT attribute indicates the next node to forward slots from B1.

<u>ENTITY TYPE</u>	<u>ENTITY NAME</u>	<u>ATTR NAME</u>	<u>ATTR VALUE</u>	<u>DESCRIPTION</u>
NODE RESOURCE B1				Resource for node representing loop interface unit 1.
		BUFFER	BUFFER1	This attribute is the name of the buffer associated with Node B1.
		NEXT	B2	This attribute is the name of the next node in the network to circulate the slot to.
		M.ROUTE	0	Delay time to route a slot. Assumed to be zero.

SEC/CHR PROCRATE Processing rate of the
node processor in sec-
onds/character.

4.3.7.2 Process: STARTUP The function of the Process STARTUP is to create slots at an origin node and initiate the circulation of them around a loop. Since each slot contains an attribute, CNODE, which contains the name of the node at which the slot is resident, the attribute is initialized to the current node with the \$CNODE keyword.

The Process STARTUP is initiated at the beginning of simulation and represents the system generation function.

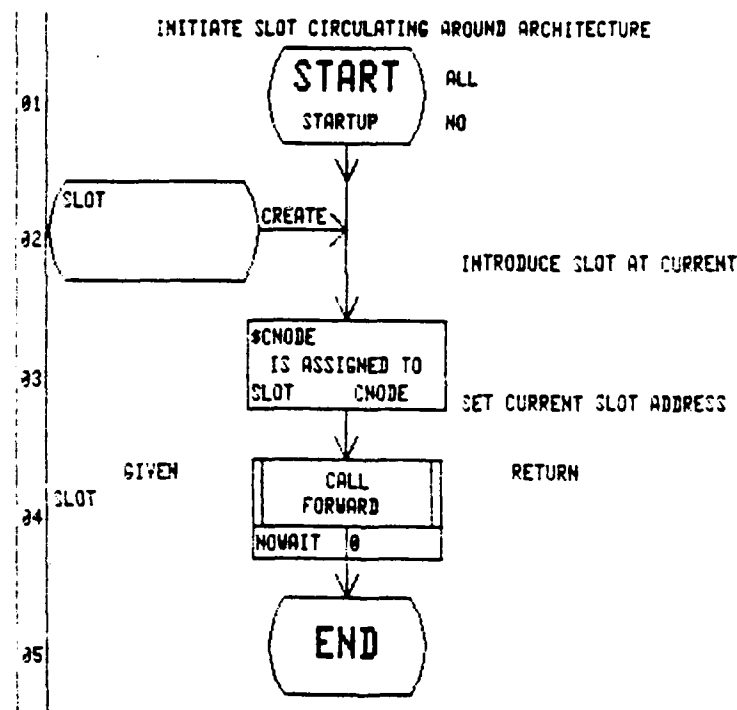


Figure 42. Example 2 Process STARTUP

4.3.7.3 Process: FORWARD Process FORWARD is a simplification of the logic for routing over the Processes used in example 1. This Process is simplified by taking advantage of the loop constraints, which are that flow from any node to another only goes in one direction to adjacent nodes. Similar logic in FORWARD is used as in CHLIO. The current node of the SLOT is determined from the SLOT attributes. The next node on the loop is determined from the NEXT attribute of the current node. The connecting channel is determined by the \$LINK keyword and the SLOT is transmitted across the channel. At the next node, the current node attribute of the SLOT is updated and the interrupt Process,

REC-SLOT, is called.

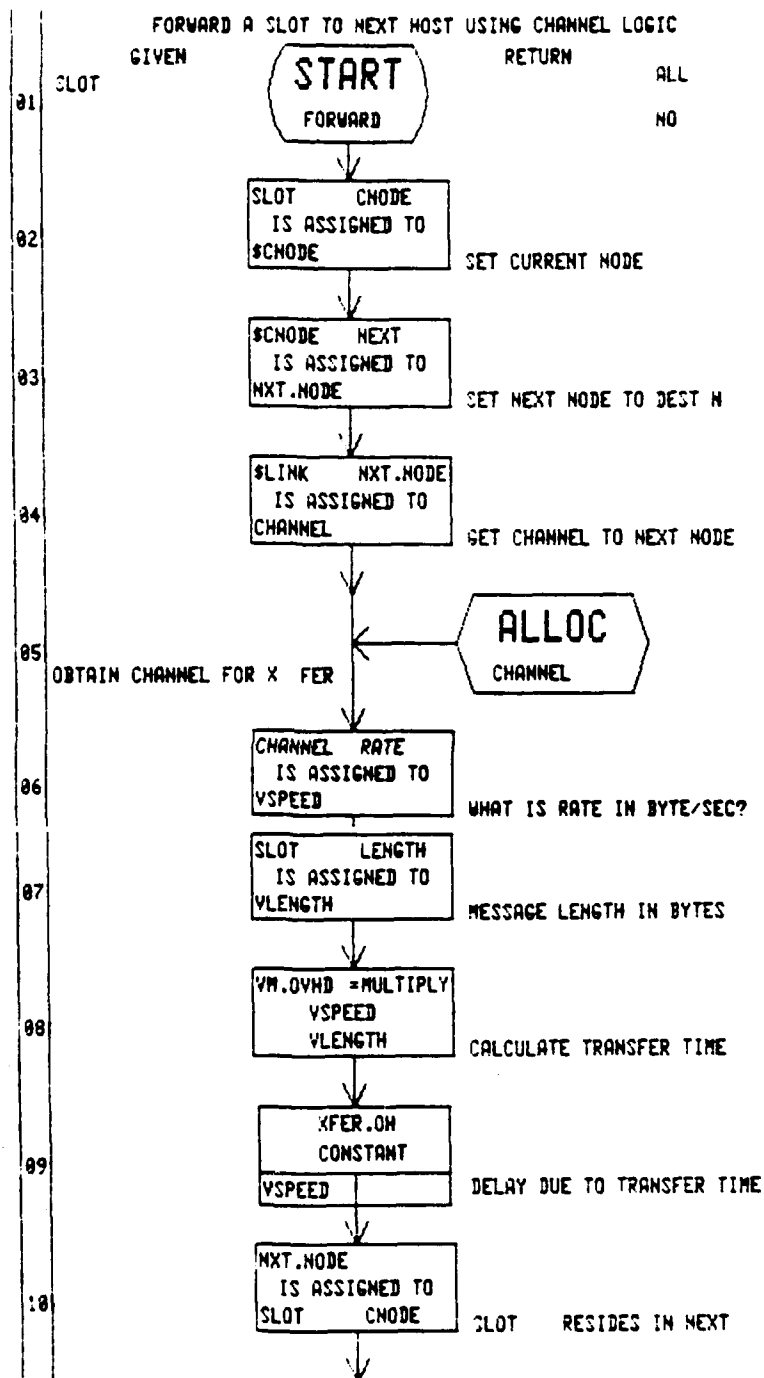


Figure 43. Example 2 Process FORWARD

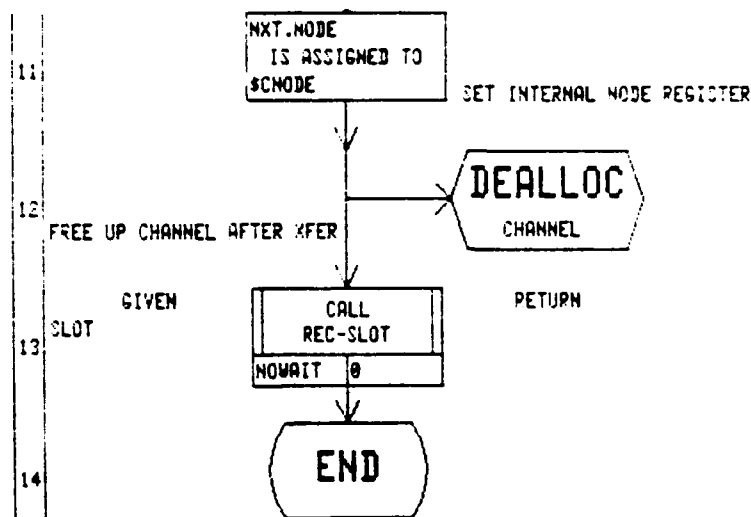


Figure 43. Example 2 Process FORWARD (cont'd)

4.3.7.4 Process: GEN-MSG Process GEN-MSG represents the activity at a host loop interface unit for message handling. Messages are introduced at the host loop interface unit and a destination node is selected by random sampling from the NODE-TBL. The buffer at the host is determined and the message is filed onto it by a first in-first out discipline. The logic of this Process is somewhat more complicated than in the instance of the Process used in the high level structure.

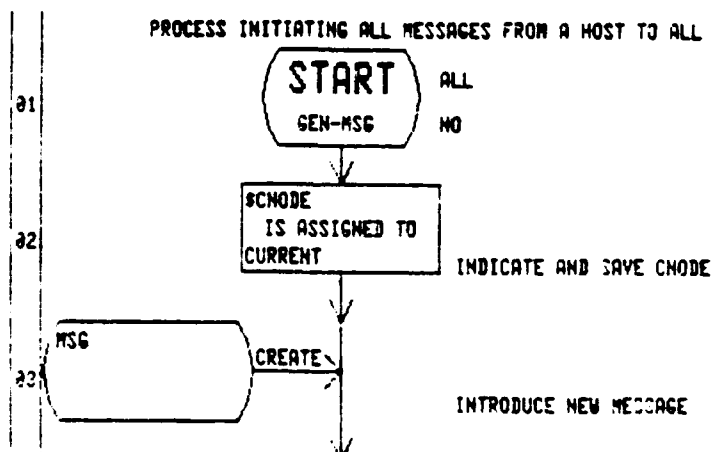


Figure 44. Example 2 Process GEN-MSG

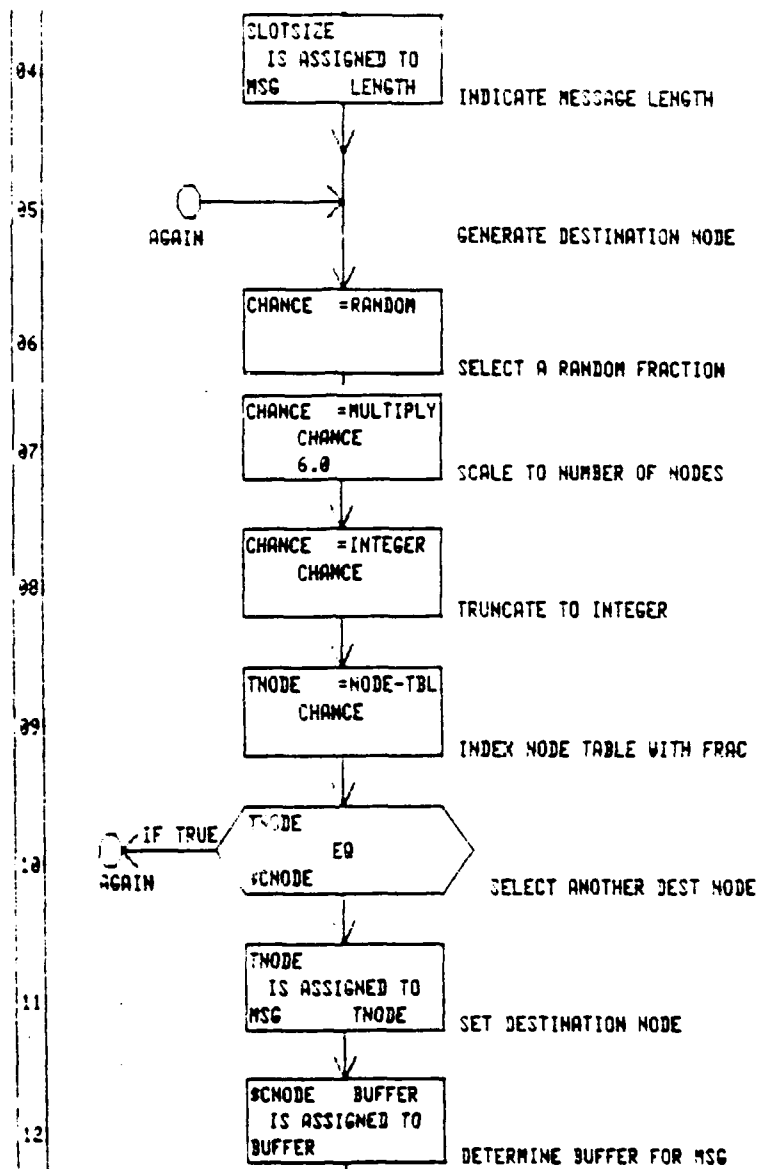


Figure 44. Example 2 Process GEN-MSG (cont'd)

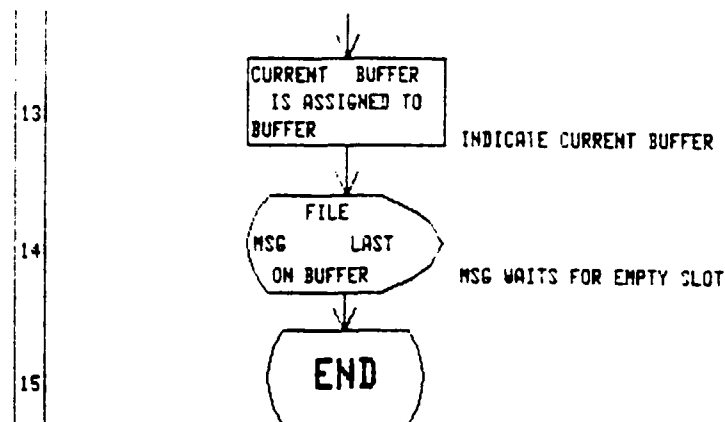


Figure 44. Example 2 Process GEN-MSG (cont'd)

4.3.7.5 Process REC-SLOT Process REC-SLOT is the most complicated of the loop system Processes because it must represent the loop interface unit logic for message handling. There are two functions which must be performed--message sending and message receiving. The logic for both functions is described in the following cases.

Case 1. SLOT Occupied - If the message is destined for the host associated with the current node, the message is removed from the SLOT and processed. The SLOT is available for inserting a message if the buffer has one. If the message is not destined for the host of the current node,

AD-A137 719

AISIM (AUTOMATED INTERACTIVE SIMULATION MODEL) TRAINING 2/2

EXAMPLES MANUAL(U) HUGHES AIRCRAFT CO FULLERTON CA

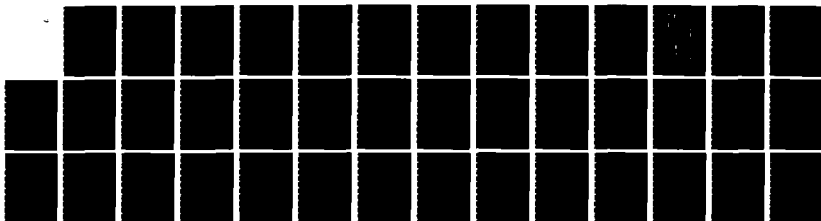
GROUND SYSTEMS GROUP M DESHLER ET AL. 26 FEB 82

UNCLASSIFIED

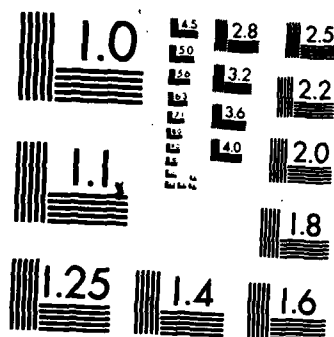
ESD-TR-83-254 F19628-79-C-0153

F/G 9/2

NL



END
PAGE
3
DTC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

the SLOT is forwarded to the next node on the loop.

Case 2. SLOT Not Occupied - If the SLOT does not have a message, one can be inserted from the buffer associated with the current node loop interface unit. In either case, the SLOT is forwarded on the loop.

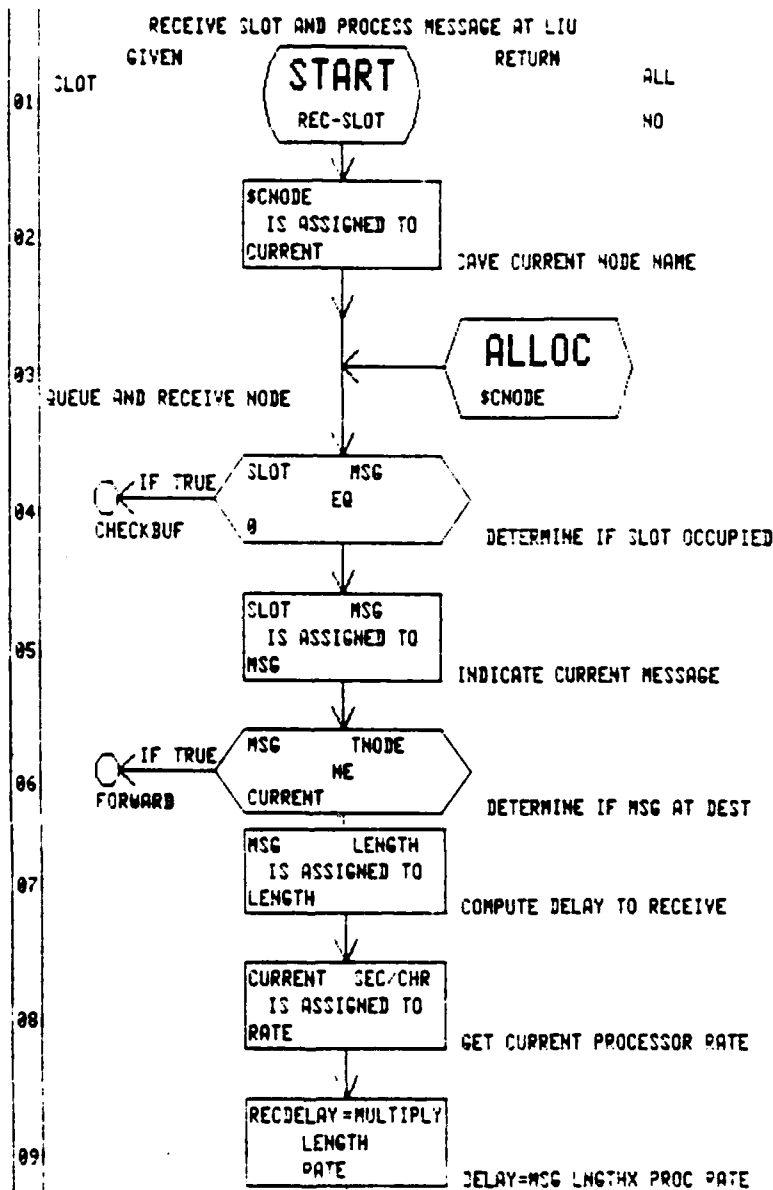


Figure 45. Example 2 Process REC-SLOT

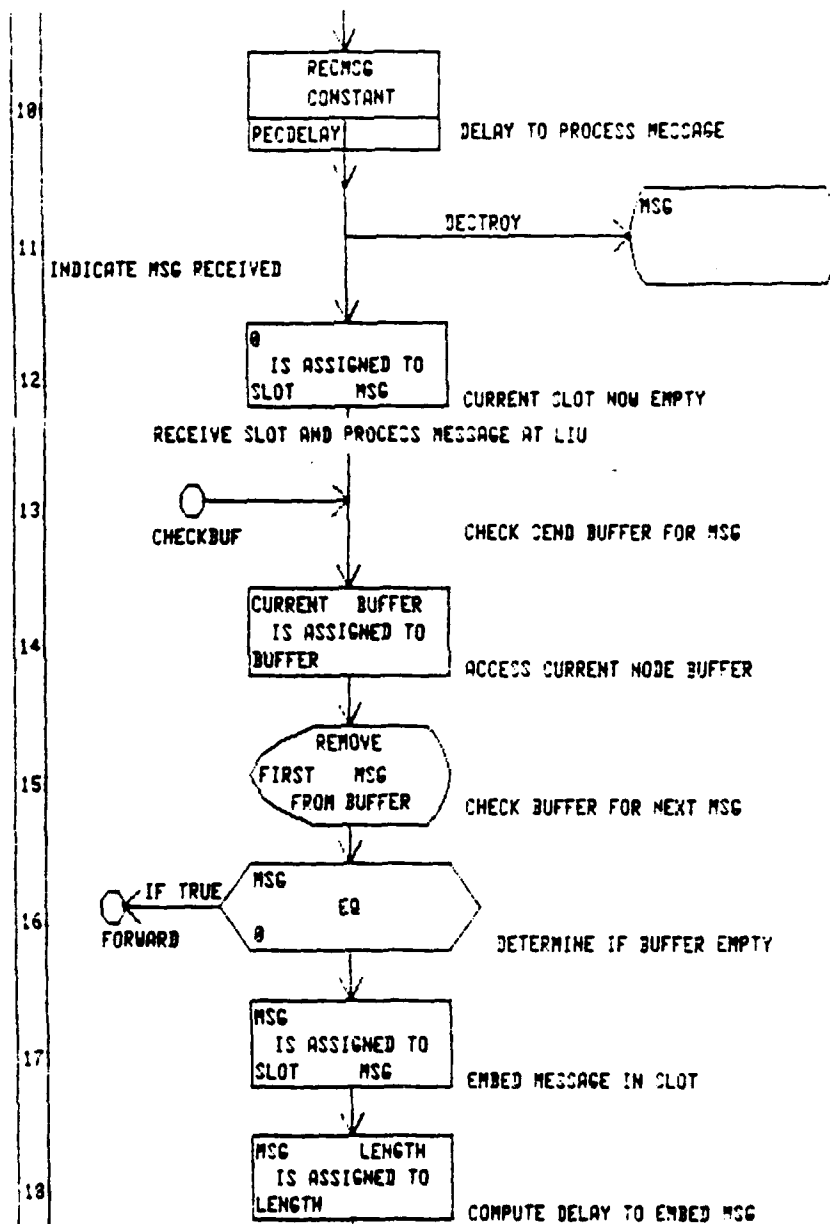


Figure 45. Example 2 Process REC-SLOT (cont'd)

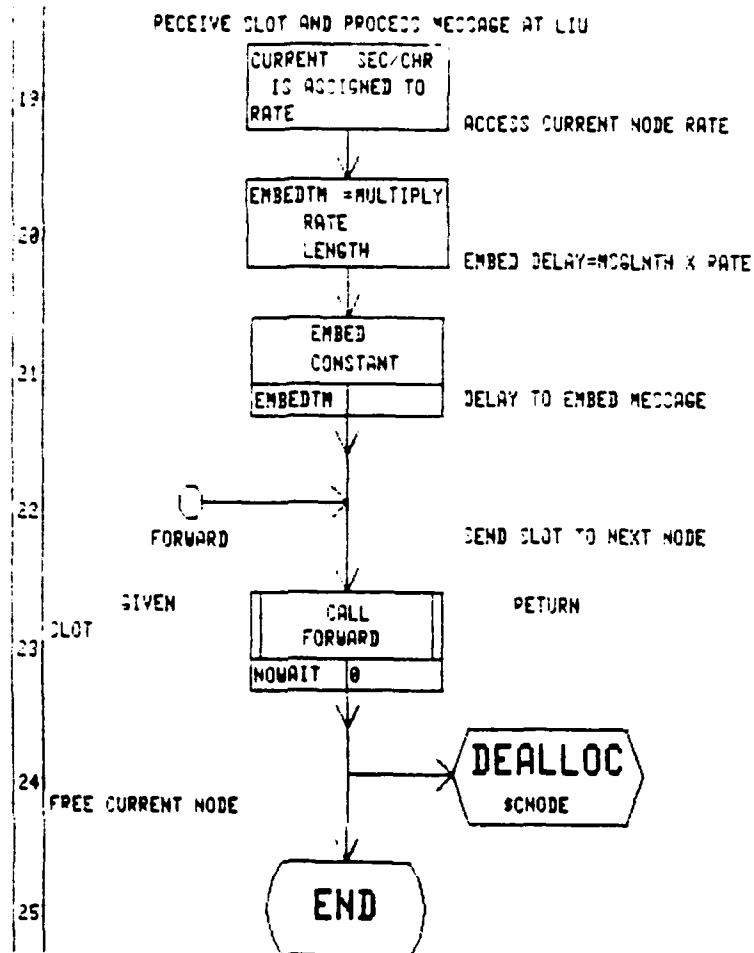


Figure 45. Example 2 Process REC-SLOT (cont'd)

4.3.7.6 Scenario and Loads

4.3.7.6.1 Load: INIFLOAD Load INIFLOAD initiates the Process STARTUP at all nodes in the architecture at the start of the simulation. This introduces the SLOTS into the system and starts them circulating.

LOAD: INIFLOAD

DESCRIPTION: START SLOTS CIRCULATING AT ALL NODES

<u>NODE1</u> B1	<u>NODE2</u> B2	<u>NODE3</u> B3	<u>NODE4</u> B4		
<u>NODE5</u> B5	<u>NODE6</u> B6	<u>NODE7</u>	<u>NODE8</u>		
<u>PROCESS</u> <u>STARTUP</u>	<u>MAX</u> ‡ 1	<u>SCH MTD</u> <u>START</u>	<u>MEAN</u>	<u>DELTA</u>	<u>PRIORITY</u> 0

Figure 46. Example 2 Load INIFLOAD

4.3.7.6.2 Load: LOADB1 Load LOADB1 represents the initiation of GEN-MSG processes at node B1. Instances of the Process are activated exponentially distributed over time around a mean time of 275 seconds. This models random generation of messages. Similar Loads are defined for all the host processors on the loop.

LOAD: LOADB1 DESCRIPTION: THIS IS THE LOAD TO GENERATE AT B1

<u>NODE1</u> B1	<u>NODE2</u>	<u>NODE3</u>	<u>NODE4</u>		
<u>NODE5</u>	<u>NODE6</u>	<u>NODE7</u>	<u>NODE8</u>		
<u>PROCESS</u> GEN-MSG	<u>MAX</u> # 100	<u>SCH MTD</u> EXPONENT	<u>MEAN</u> 275	<u>DELTA</u>	<u>PRIORITY</u> 0

Figure 47. Example 2 Load LOADB1

4.4 Analysis of Results

4.4.1 Performance Measures The B nodes in the architecture represent loop interface unit processors. The d nodes represent host processors. The channels are modeled by resources such as B1B2 which represents the channel between the loop interface unit for host 1 and the loop interface unit for host 2. All other channels are similarly named. The performance statistics for these entities are found in the AISIM Resource Report.

4.4.1.1 Queueing Time Queueing time, described in section 4.1.2 as the elapsed time from message generation until placement on the loop can not be obtained from the model using the initial approach without slots and buffers. The motivation for extending the model was to obtain this statistic. In the extended model with slots and buffers, the simulation results for queueing time are found in the Queue Report. BUFFER1 through BUFFER6 represent the loop interface unit buffers for units 1 through 6 respectively. The TIME IN QUEUE heading lists the MEAN, STANDARD DEVIATION, MINIMUM and MAXIMUM times for the number of samples taken under TOTAL NUMBER REMOVED.

4.4.1.2 Transmission Time The total message transmission time, which includes queueing delays and transmit times, is found in the Item Report. The Item Report lists the MINIMUM, MAXIMUM, MEAN and STANDARD DEVIATION for the TIME IN SYSTEM for messages. This is accumulated under the Item name MSG. The number of samples in the statistics are based on the NUMBER DESTROYED. Messages which have not been destroyed at the end of the simulation are not counted. Both the initial model and the extended model produce this statistic. Because the extended model with slots and buffers represented is at a more detailed level, the transmission time statistic is more accurate.

The loop transmission time, defined in section 4.1.2 as the time elapsed from the start of message placement on the loop until the last character is received and removed from the loop, is not computed directly by the model. No structure corresponding to this has been built into the model. In order to get this result, it is necessary to eliminate queueing and processing delays from the total time in system for each message. This can be done by hand calculation if this statistic is required although total transmission time appears to be a more important result.

4.4.2 Validating the Model Some validity for the model can be obtained by comparing analytic expected utilizations of nodes and channels with simulation produced results. Since the Load was defined equally for all source nodes of messages, and destination nodes were selected by a uniform distribution, results for all channels on the loop and loop interface units are very close. Analytically derived utilization for the loop interface units is 2.91%. This is correlated closely by the AISIM MEAN # BUSY statistic for Resources B1 through B6 which vary from 0.029 to 0.033. Utilization of channels is computed analytically as 30.54%. The MEAN # BUSY statistic for Resources B1B2, B2B3, B3B4, B4B5, B5B6 and B6B1 average 0.30.

4.5 Conclusions

The loop communication system shown in this example is very effective in demonstrating some key concepts of AISIM simulation analysis. Using much of the logic from the previous example, it was shown how a model of a totally different system could be

constructed very rapidly. The high level model could easily be built in one interactive session. Simulation results for the system could be obtained within a matter of hours. The high level model represented the architecture, connectivity and Load on the system.

In order to gain data on the lower level elements of the system, the logic of the model was extended to model SLOTS and BUFFERS. This also could be accomplished rapidly once the structure of the model is laid out.

Techniques used in this example for random sampling from a number of different nodes using a table and using the attributes of Resources and Items to hold data on relationships are applicable to many modelling problems.

5. Bus Communication System Example

This example uses bus technology as the communications medium. It is representative of a class of systems that deal with multiple hosts communicating over a network. This example is intended to study the design characteristics which are basic to this type of communication system.

5.1 Input

5.1.1 Mission Concept Data is distributed among many hosts in a network. There is a frequent need for a host to access data resident in another host's files. Rapid communication between hosts is effected by a digital communication network which controls communication and transmits data between the hosts on demand. Full disk track transfers are typical of the internetwork communication traffic.

5.1.2 Problem Perspective A study of the following characteristics of bus communications is basic to this system:

1. Bus capacity
2. Processor capabilities
3. Bus Protocols
4. Buffer Requirements

The load on this system can be viewed as message traffic which is the incidence of data requests generated at the hosts. Random requests would represent a worst case study for a specific inter-generation rate. Data request destinations from each host can be distributed among the hosts.

The following measures assess the performance of the communication system.

1. **Response Time.** Response time is the time from the end of sending the message to the start of receiving the response. It is the user's wait time after sending a message and before receiving a reply. Included in response time are: 1) the communication time between the end of the input message and its receipt by the destination processor, 2) the processing time before transmission of a reply, and 3) the communication time between the initiation of the reply message and its receipt by the user.
2. **Communication Time.** Communication time refers to the time required to transmit the message on the bus. Specifically, it is the time between end of sending the message by the originating BIU to end of receipt of message by the destination processor, plus the time between start of reply from

the processor to start of receipt at the originating BIU. Communication time is the same as the response time with the processing time removed.

3. Round Trip Time. Round trip time measures the total duration of message transmission and processing from start of sending the input message to end of receiving the reply message. It is the user's total cycle time from the time he first starts sending until he has received the entire reply.

5.1.3 System Description The interhost communications network consists of two time division, multiple access (TDMA) buses. A control bus contains the necessary information to control the internost traffic. Data transfers utilize the other bus, the data bus.

The basic elements of the TDMA bus system are

1. the transmission medium, i.e. the buses;
2. the bus interface units (BIUS) which are the elements which interface each device with the bus;
3. the element which provides central control of bus operations, i.e. the network control element.

Each bus consists of a serial data stream. Information is time divisioned multiplexed onto the bus by arranging it in intervals called time slots. The total capacity of the control bus is shared simultaneously by the hosts and the network control element by allocating sets of time slots dynamically. The time slots are sequentially accessed. Each host plus the network control element is assigned a time slot. A complete cycle of slots is called a frame. A portion of the control bus is allocated to each host and the network control element for message transmission.

The data bus is divided into fixed slots of 512 bits. The slots on the data bus are not preallocated. The entire data bus is allocated for a given transmission between hosts.

The message communication protocol for communications between hosts is described by the following sequence.

1. Host1 places a "Data Request" message in a queue for a control message output buffer in the Host1 BIU.
2. When a buffer is available, the request message is queued for the channel and then transmitted to the BIU.
3. The BIU places the request message in a preassigned slot on the control bus and the message is transmitted to the Host2 BIU.

4. The Host2 BIU queues the request message for transfer to the Host2 processor.
5. The Host2 BIU sends a message acknowledgment to the Host1 BIU on the control bus.
6. Host2 processes the request message (50 ms processing time).
7. The Host2 readies the data message and queues it for the BIU data output buffers (2).
8. When the buffers are available, the first portion of the message is queued and transferred to the BIU.
9. The BIU sends a "Bus Request" message to the NCE over the control bus requesting use of the data bus.
10. The request is queued and processed by the NCE, which assigns the data input buffers of the Host1 BIU to the reply BIU when they become available.
11. When step 10 is completed and the data bus becomes available, the NCE sends a "Bus Grant" message on the data bus to the Host2 BIU.
12. Upon receipt of the grant, the Host2 BIU begins sending the message (448 data bits per slot) to the Host1 BIU over the data bus. The BIU buffers are toggled at both the sending and receiving BIUs.
13. When the data message transfer has been completed, the Host1 BIU sends an "Acknowledgment" message to the Host2 BIU and a "Bus Release" message to the NCE. Both messages are sent over the data bus. The Host2 then releases its BIU data output buffers.
14. The NCE queues and processes the "Bus Release" message and releases the data bus for other users.
15. Parallel with step 12, the Host1 BIU queues and transfers each buffer of the message to its host.
16. When the last Host1 BIU input buffer has been transferred to its host, the BIU sends a message on the control bus to the NCE telling it to release its data input buffers.
17. The NCE queues and processes this message and releases the Host1 data input buffers.

5.2 Preliminary Analysis

5.2.1 Justifying AISIM Simulation The characteristics of the bus system map into AISIM entities.

1. Procedural Operations - The sequence for transmitting messages and data between hosts is described in the system description. This is a structured, well defined sequence which is easily modelled by procedural logic.
2. Parallel Processing - The host processors, bus interface units, data bus and network control element function in parallel.
3. Resource Sharing - The data bus and network control element are shared between all hosts through bus interface units. The sharing of these elements are governed by the division, multiple access control.
4. External Loading - The loading on the network is described by message traffic characterized by a message generation rate at each host. Messages are distributed to all hosts.

5.2.2 Problem Definition The most important aspects of the bus communication system to model is the interface between the hosts, the bus interface units, the data bus and the network control element. All these elements in the system will be represented. The logic of each device is defined by the bus protocol. This protocol will be represented as accurately as it is described in the problem statement. This includes data requests, acknowledge and control messages.

Modelling the architecture of the network is of less importance because the network control element and data bus effect direct connections between all devices through the bus interface units.

The physical characteristics of the devices will be parameterized by variables. This applies specifically to the band width of the buses, the processing rate of the hosts and the processing times for messages at the network control element.

Queueing and utilization statistics will be generated on the host processors, channels between hosts and interface units, control bus, data bus and the network control element.

Simulation runs define a distribution of messages to be loaded into the system. Messages are distributed by source. This is described by matrices in the accompanying figure.

5.2.3 Definition of Objective The objective of modelling the bus communication network is to represent the logic of the bus protocol accurately and quantify the performance of the communication based on varying load mixes.

5.3 Model Build

5.3.1 Design, Plan and Construction of the Model The seven most case will be represented as the system for the purposes of demonstrating this example.

5.3.1.1 Model Design All Processes in the model represent message handling functions. For this reason, Item MSG will represent all message types. The first step in the design is to determine the characteristic attributes of the Item MSG. The attributes of MSG will play a big part in the definition of the model Processes.

Every message is described by the following set of attributes:

<u>Attribute Name</u>	<u>Attribute Description</u>
CNODE	The current node in the bus network which is processing the message.
DEST	The destination node of the message.
LENGTH	The number of characters in the message.
ORIGIN	The source node of the message in the bus network.
TYPE	The type of the message, one of the following: data request, data message, acknowledge message, bus request, or bus release.

The top level structure of the model makes up the initial model design. The high level functions of the bus system are assigned to AISIM entities Scenario, Loads, and Processes.

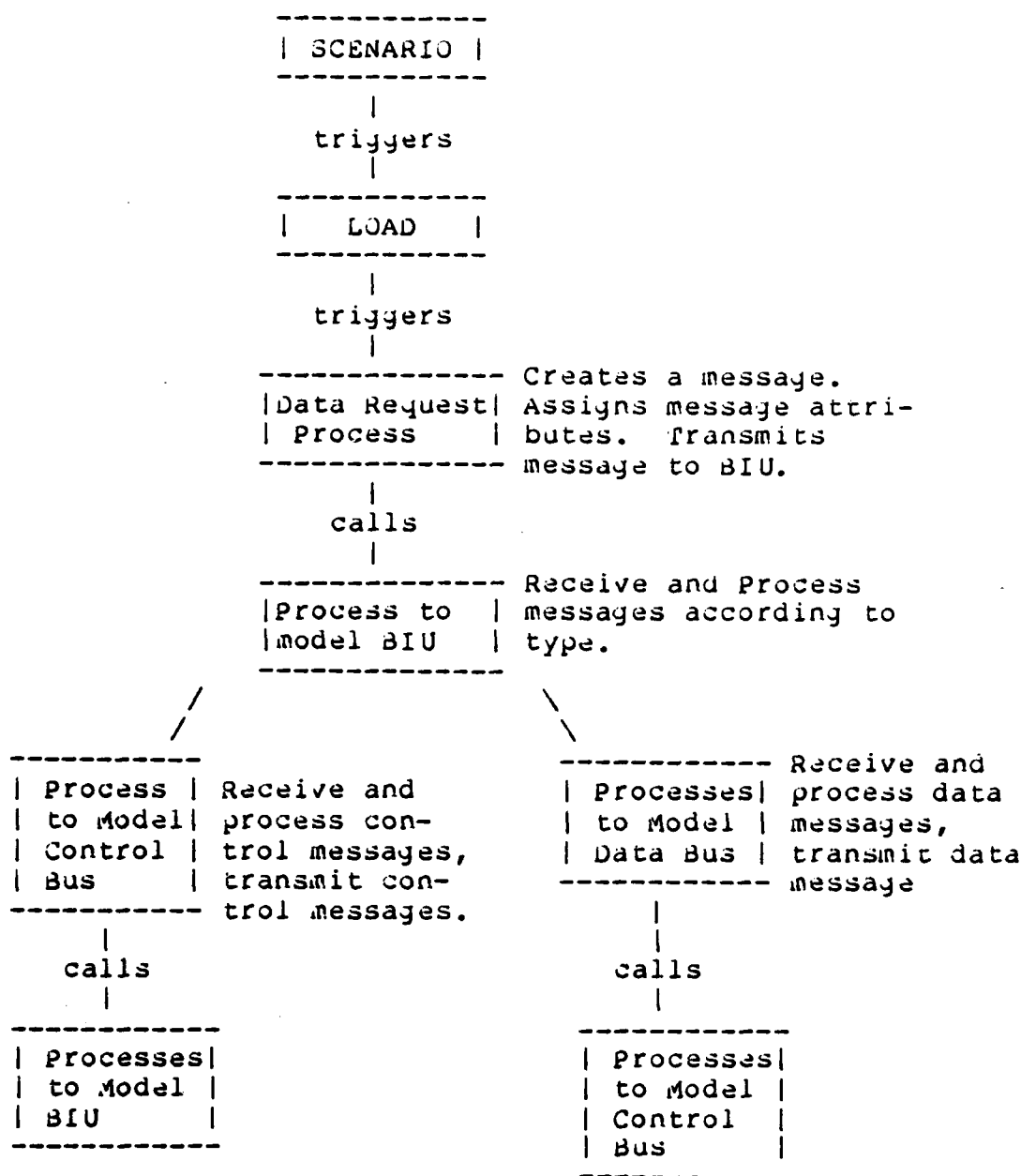


Figure 50. Example 3 high Level Structure

5.3.1.2 Model Implementation Plan Since the Architecture of the bus system is fairly simple, it is not a good idea to use the message routing submodel developed in example 1 which is designed for complicated networks. The bus communication system is considerably different from the loop system described in example 2 so it is not probable that any of the component models from example 1 and example 2 can be used for example. In this case it is

necessary to build the bus system model from scratch.

The most critical element of the bus system is the protocol between the devices, the bus interface units, the network control element and the data bus. Priority is placed on building the Processes modeling the control logic of these devices first in the plan.

The sequence for constructing the model of the bus communication network is described in the following steps.

1. Design and construct Processes to model the bus interface units.
2. Design and construct Processes to model the network control bus. Integrate Processes to the bus interface units.
3. Design and construct Processes to model the data bus. Integrate Process into the network control bus model.
4. Verify model on subset of architecture. Define single thread Scenario.
5. Build complete network architecture.
6. Define full loading and Scenario.
7. Verify complete network model.
8. Run simulations and analyze results.

5.3.1.3 Process: BIU The Process BIU models the logic of the bus interface units for all devices connected to the bus. The Process BIU handles all message types and performs a different sequence of logic for each type. There are five different message types handled at the bus interface unit.

1. Outgoing data request messages.
2. Incoming data request messages.
3. Bus grant messages.
4. Data messages.
5. Acknowledge messages.

5.3.1.3.1 Outgoing Data Request Message Handling For outgoing data request messages, the bus interface unit queues the message for the network control bus by placing the message in an assigned slot of the bus. This logic can be represented by passing the message to the Process representing the control bus after a delay for slot access. The AISIM primitives to do this are shown in

the accompanying figure.

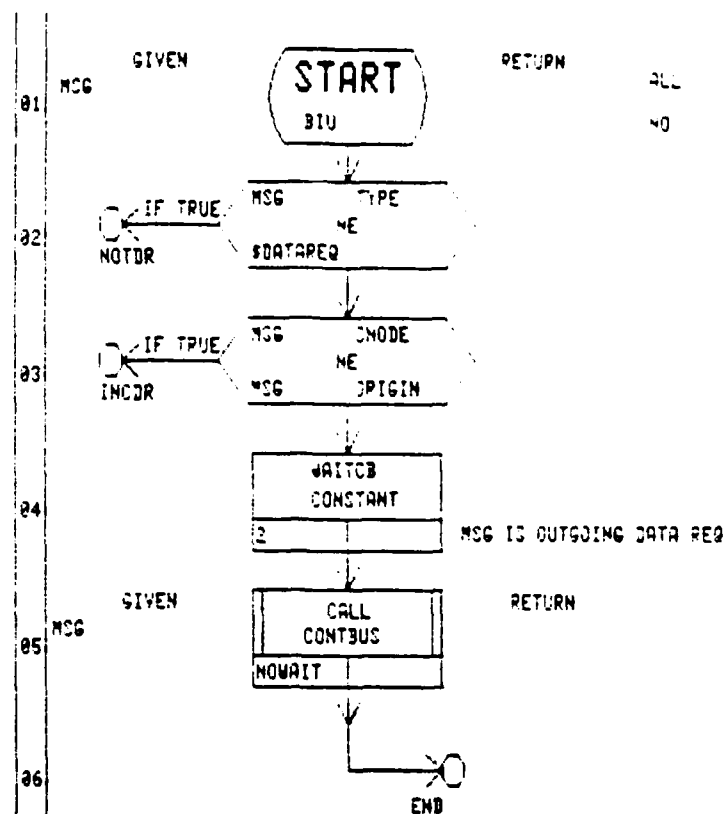


Figure 51. Example 3 Bus Interface Unit Logic For Outgoing Data Request Message

5.3.1.3.2 Incoming Data Request Message Handling For incoming data request messages, an acknowledge message is sent to the requesting host via the control bus. This is modelled by creating a second message, originating at the bus interface unit and destined for the host requesting data. A Process SENDACK is called which will assign attributes for acknowledge messages and transmit the message back by calling the Process representing the control bus logic. The data request message is transmitted from the bus interface unit to the host over the channel between these elements. A new message is created by the bus interface unit in order to request the bus. This message is placed in the slot for the control bus. The AISIM primitives for this logic are shown in the accompanying figure.

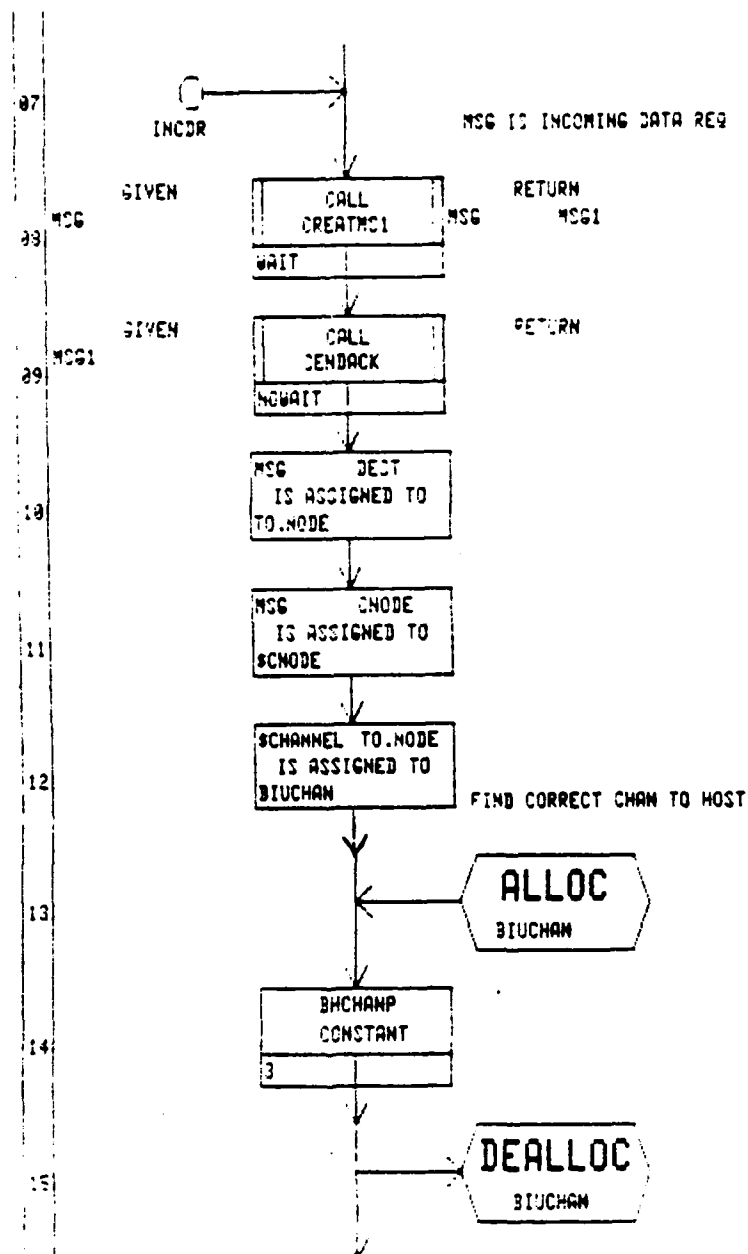


Figure 52. Example 3 Bus Interface Logic For Incoming Data Request Messages

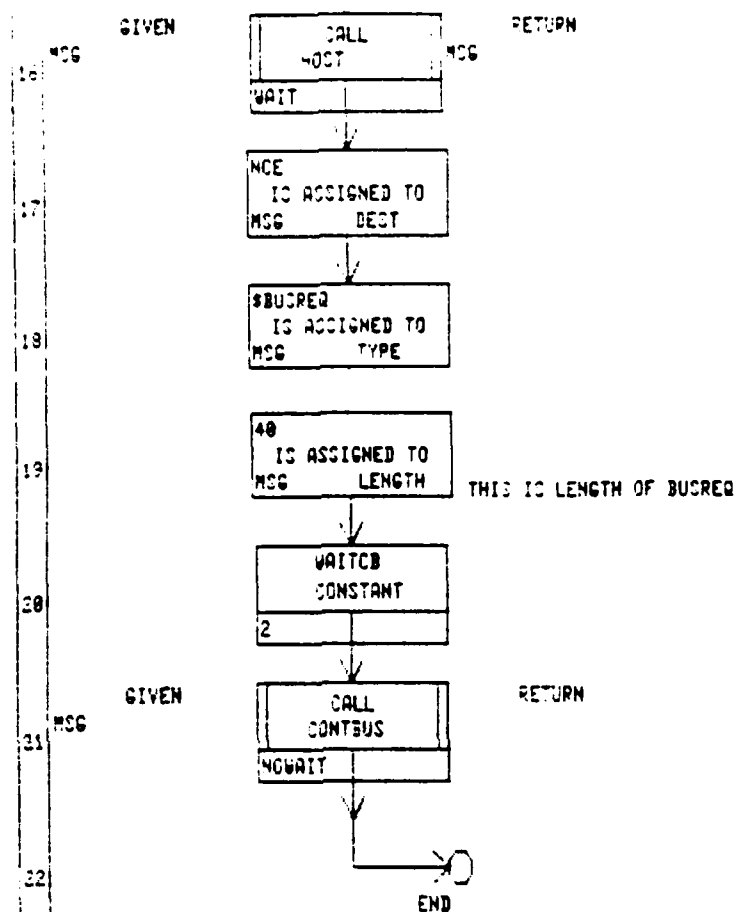


Figure 52. Example 3 Bus Interface Logic For Incoming Data Request Messages (cont'd)

5.3.1.3.3 Bus Grant Message Handling A message from the network control bus to the bus interface unit indicating a bus grant enables a data message to be sent over the data bus. The attributes of the message are assigned to indicate the origin and destination of the data message. It is reasonable to determine the message length at this time since it is randomly generated. The length is assigned to the data message and it is passed to the process representing the data bus logic.

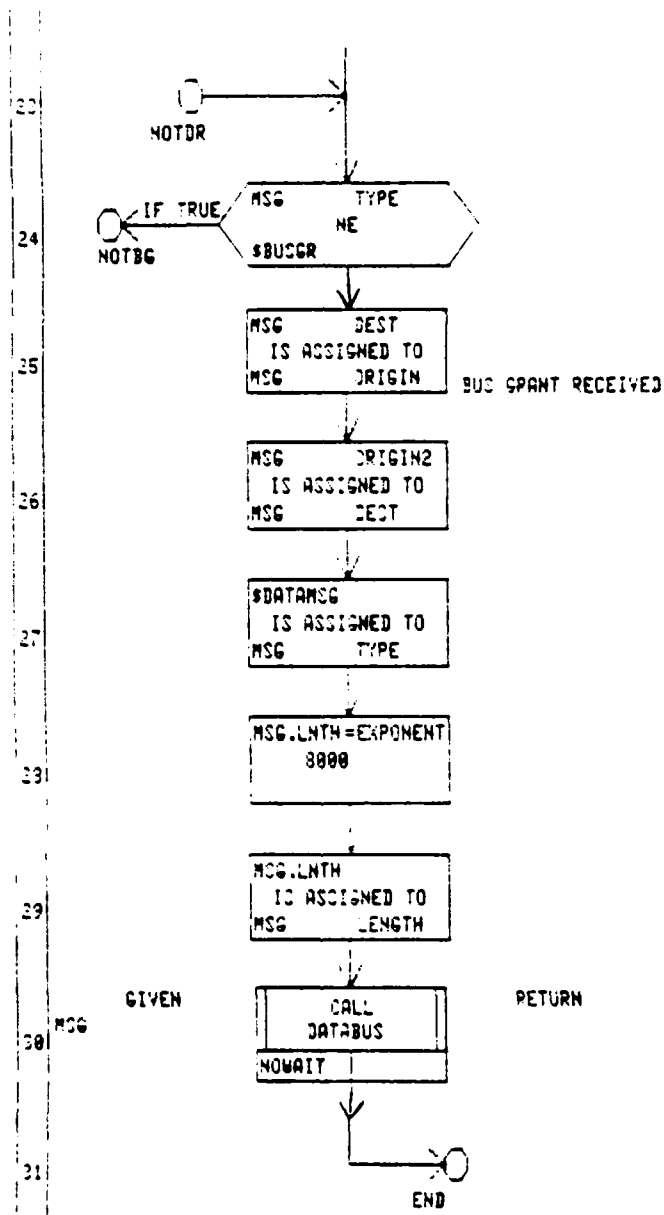


Figure 53. Example 3 Bus Interface Logic For Bus Grant Messages

5.3.1.3.4 Data Message Handling When a data message is received at a bus interface unit, an acknowledge message is created and sent via the control bus back to the host which sent the data. Another message is sent to the control bus to indicate that the data message is received and the data bus can be released. The data is transmitted from the bus interface unit to the host over the connecting channel. A message is then generated to cause the data message origin host to release buffers.

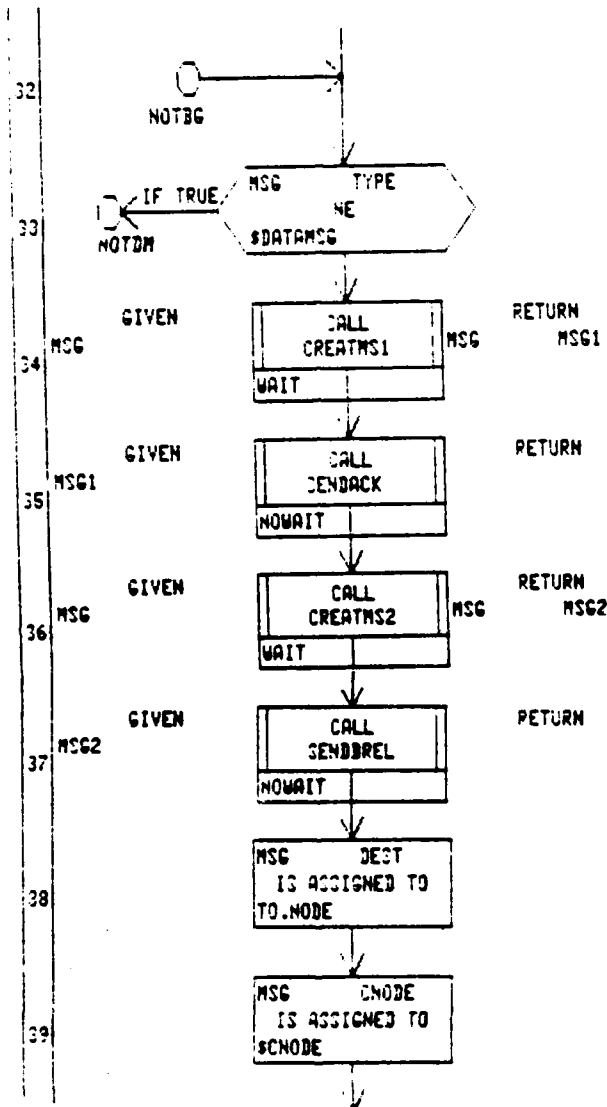


Figure 54. Example 3 Bus Interface Logic For Data Message Handling

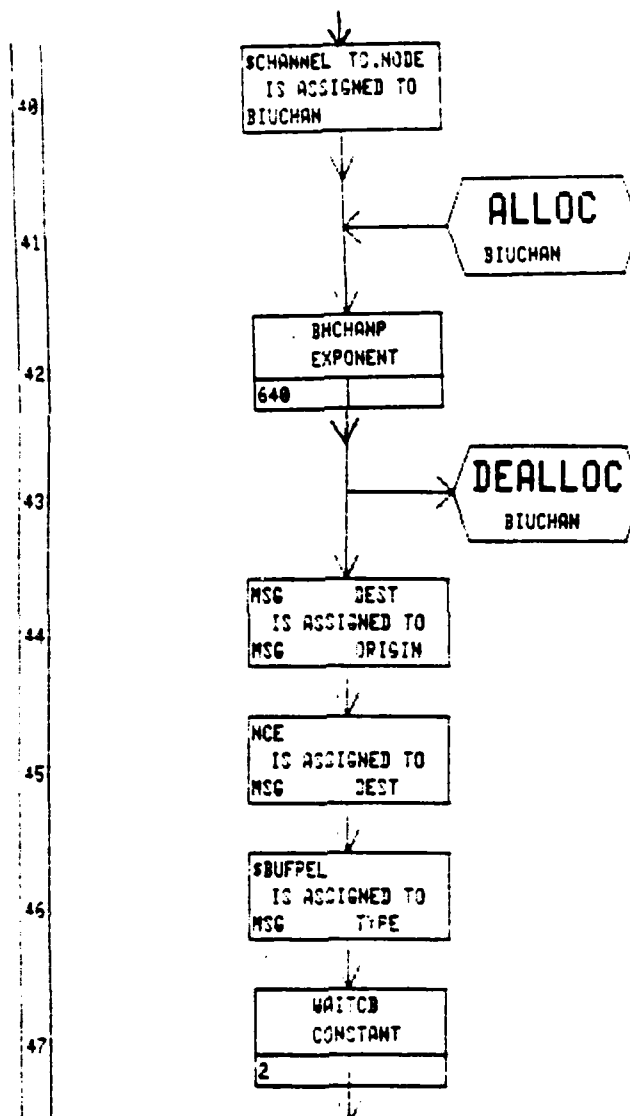


Figure 54. Example 3 Bus Interface Logic For Data Message Handling (cont'd)

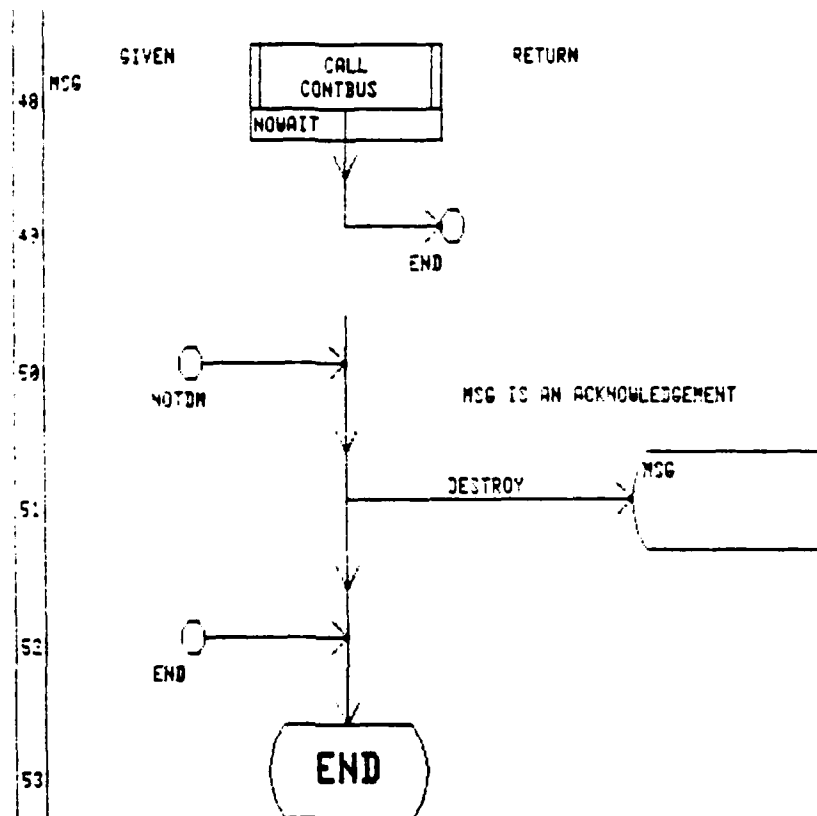


Figure 54. Example 3 Bus Interface Logic For Data Message handling (cont'd)

5.3.1.4 Process: CONTBUS The Process CONTBUS models the control bus logic which is initiated by a receipt of a message. If the message is a control message to be processed by the control bus, then the Process NCEPROC is called given the message. NCEPROC models the logic of the network control. If the message is to another device, then the message is passed to the bus interface unit handler, Process BIU, after setting the current node of the message attribute to that device bus interface unit.

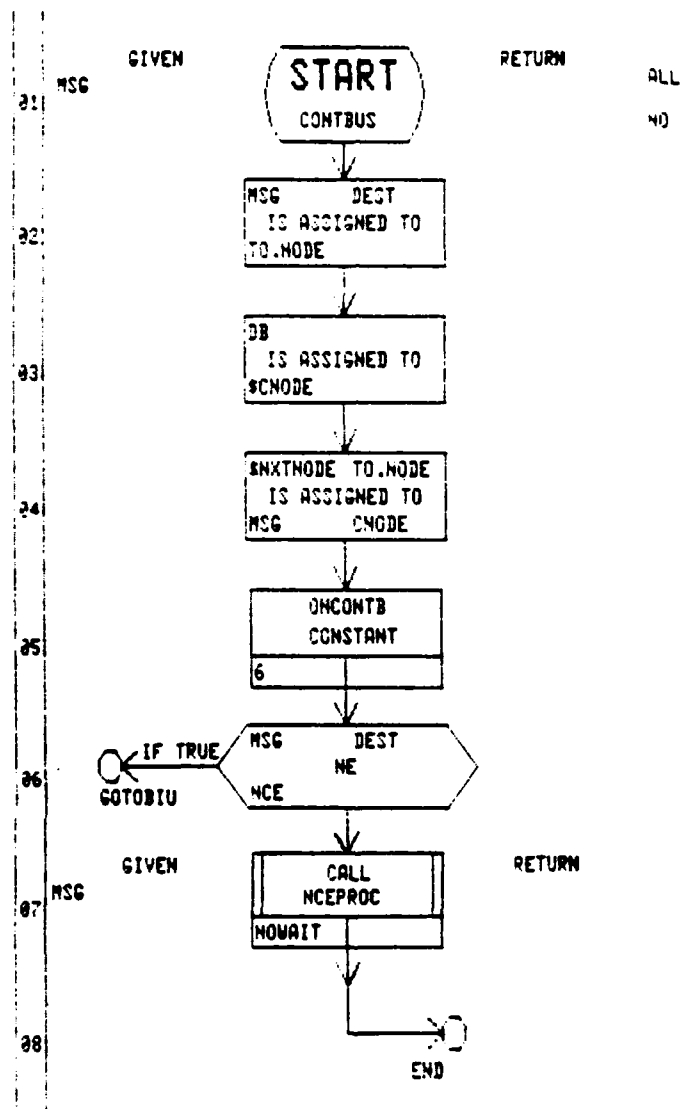


Figure 55. Example 3 Control Bus Message Handling Logic

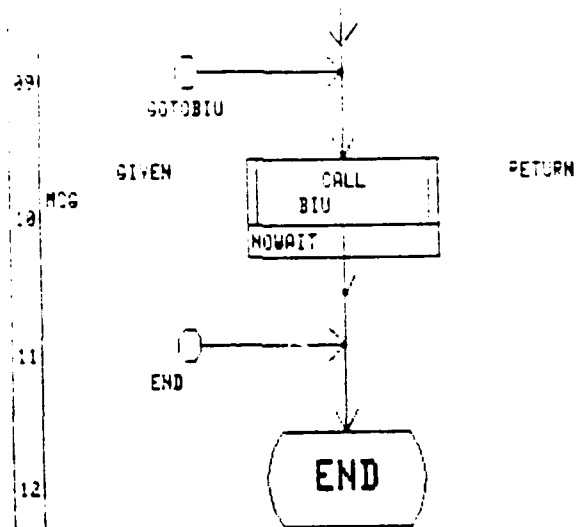


Figure 55. Example 3 Control Bus Message Handling Logic

5.3.1.5 Process: NCEPROC The network control element handles two types of messages:

1. Bus request messages
2. Bus release messages

For bus request messages, buffer space at the destination buffer interface unit is reserved, the data bus is allocated and a buffer grant message is given to the data bus. The data bus and buffers remain allocated until a bus release is received at the control bus from the receiving bus interface unit. This is implemented by the SUSPEND.

For bus release messages, the previous instance of NCEPROC for the bus request is resumed so that the bus and buffer space is released. The message is then destroyed.

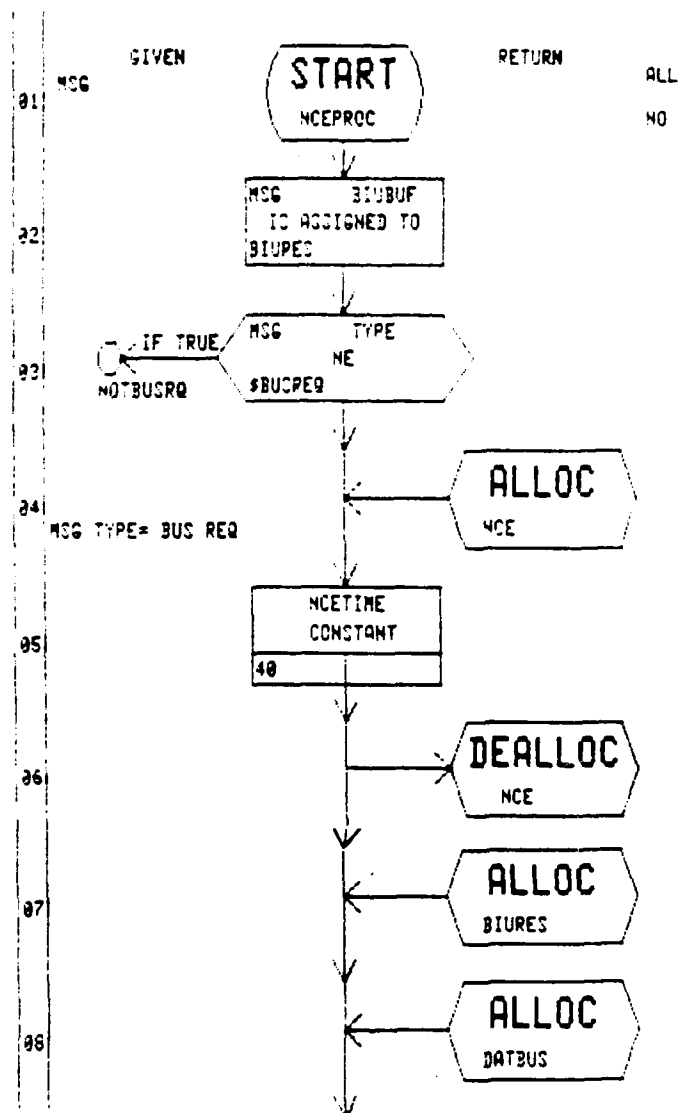


Figure 56. Example 3 Network Control Logic

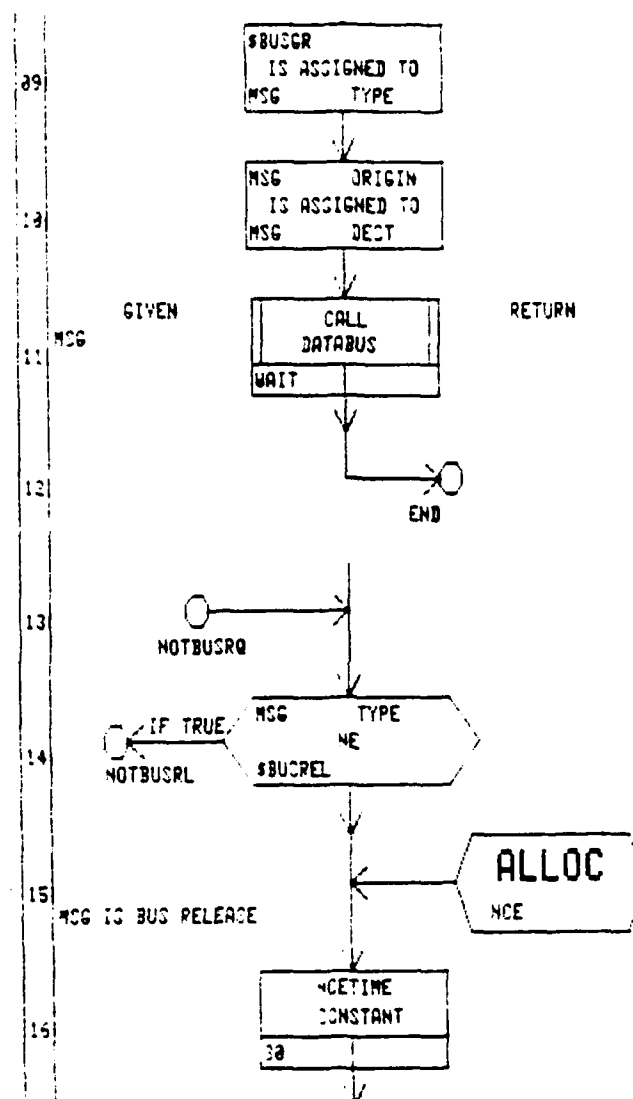


Figure 56. Example 3 Network Control Logic (cont'd)

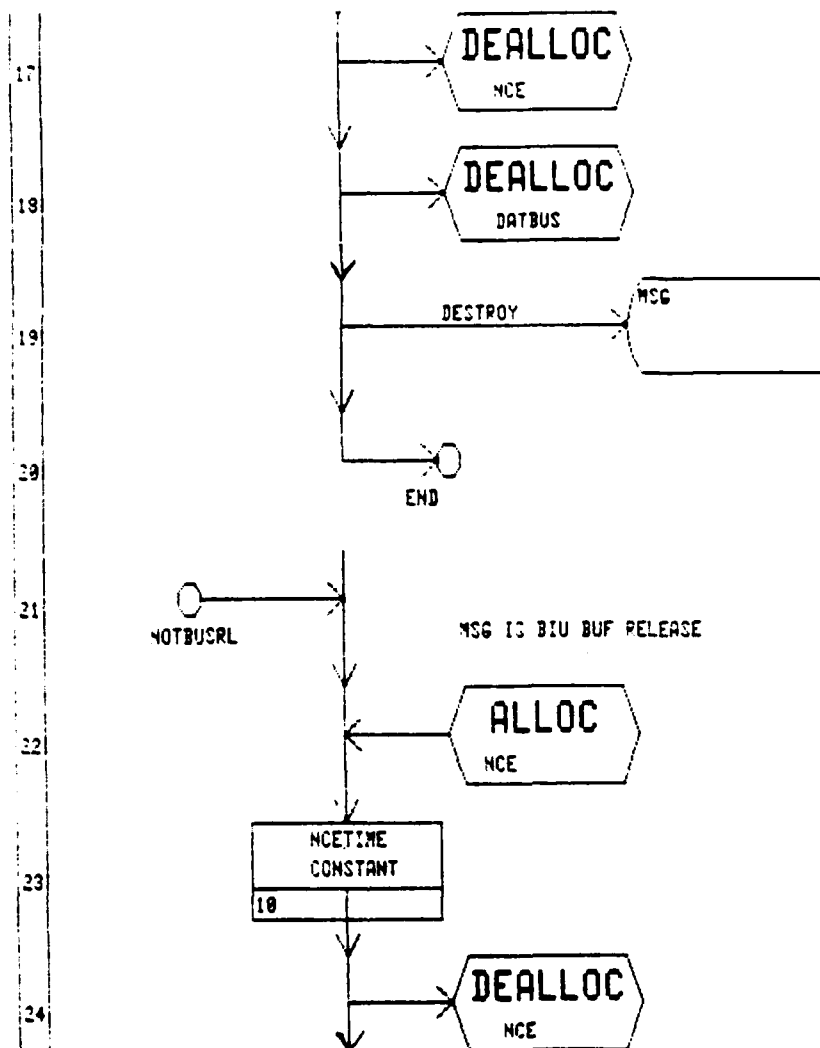


Figure 56. Example 3 Network Control Logic (cont'd)

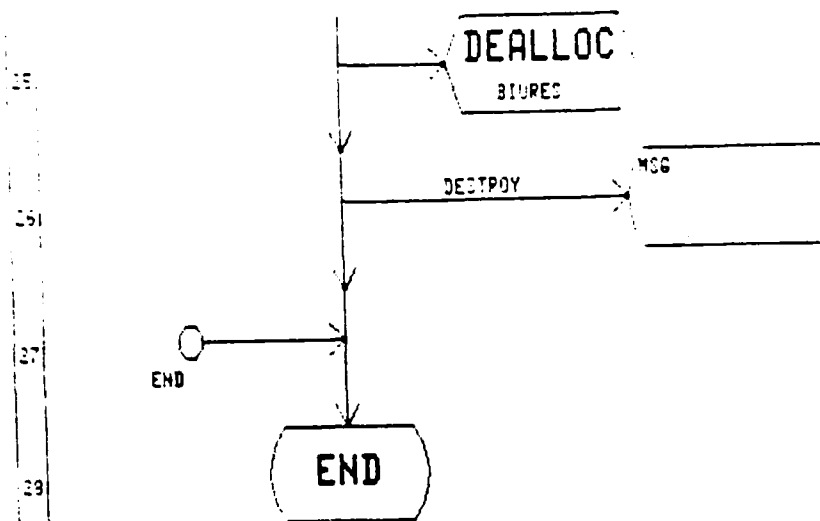
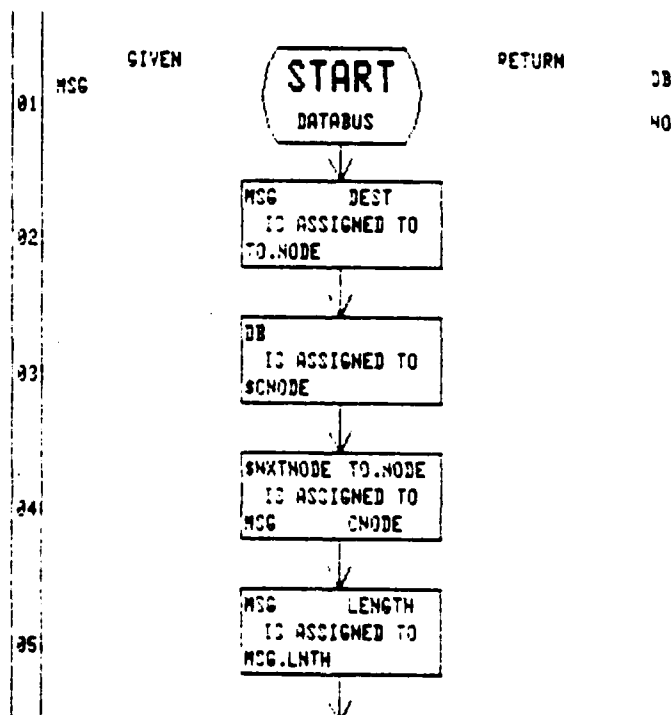


Figure 56. Example 3 Network Control Logic (cont'd)

5.3.1.6 Process: DATABUS The Process DATABUS represents the transmission of data from one bus interface unit's output data buffers to another bus interface unit's input data buffers. The delay time for this transfer is computed based on the length of the data message and the speed of the bus. When the data is transferred, the message is given to the destination host's bus interface unit if the destination is a host, or to the network control element.



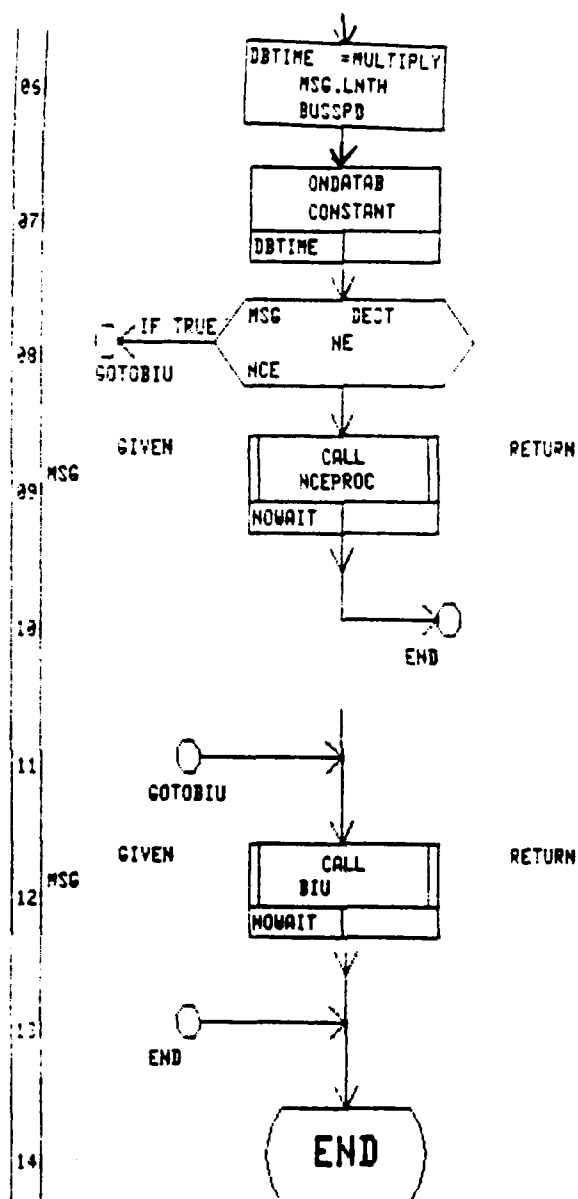


Figure 57. Example 3 Data Bus Message Handling Logic

5.3.1.7 Process: SENDACK The Process SENDACK is initiated at a bus interface unit to transmit an acknowledgement of a receipt of a data message from a receiving interface unit to the transmitting host. The acknowledge message is transmitted on the data bus.

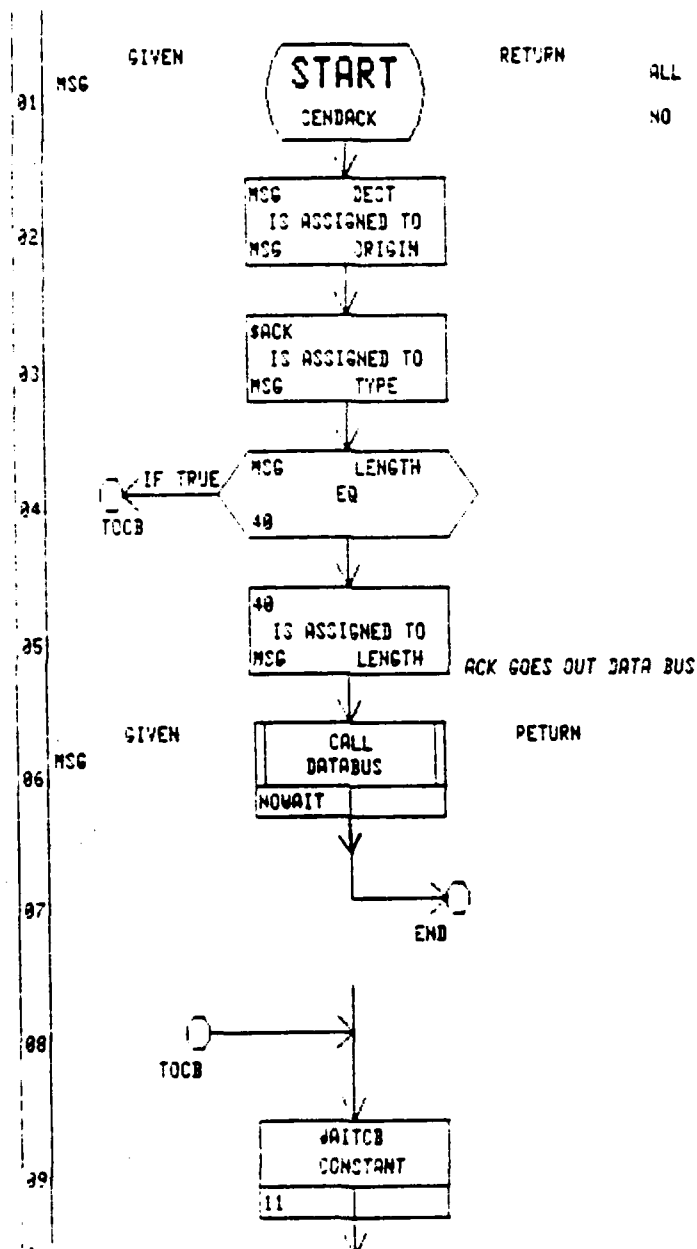


Figure 58. Example 3 Acknowledge Message Generation

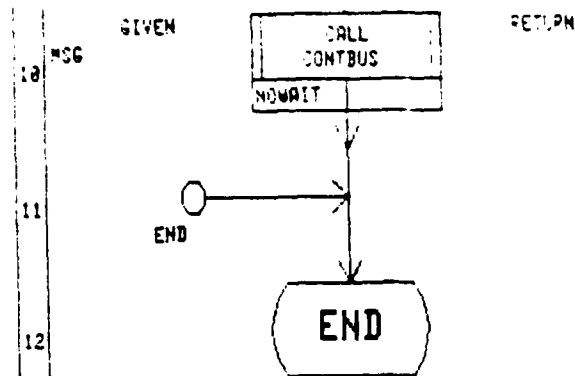


Figure 58. Example 3 Acknowledge Message Generation (cont'd)

5.3.1.8 Process: SENDREL The Process SENDREL is initiated at a bus interface unit to transmit a bus release message to the network control element so that the data bus can be freed.

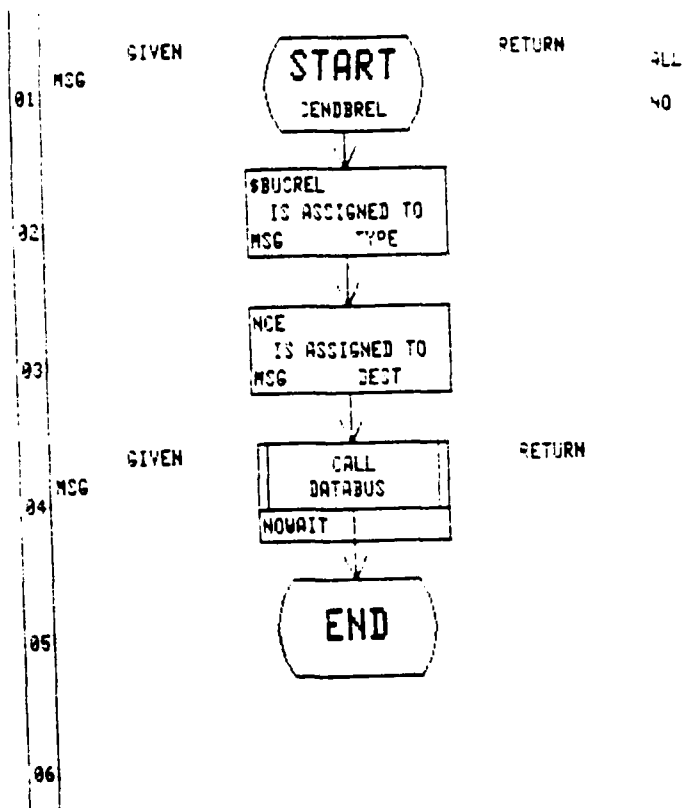


Figure 59. Example 3 Release Data Bus

5.3.1.9 Process: HOST The Process HOST represents the processing upon receipt at a host node of a data request message. The Host processor formats a data message and sends the data to the bus interface unit over the channel between the two devices. This Process is called from the Process BIU which waits after passing a data request message for the data to return.

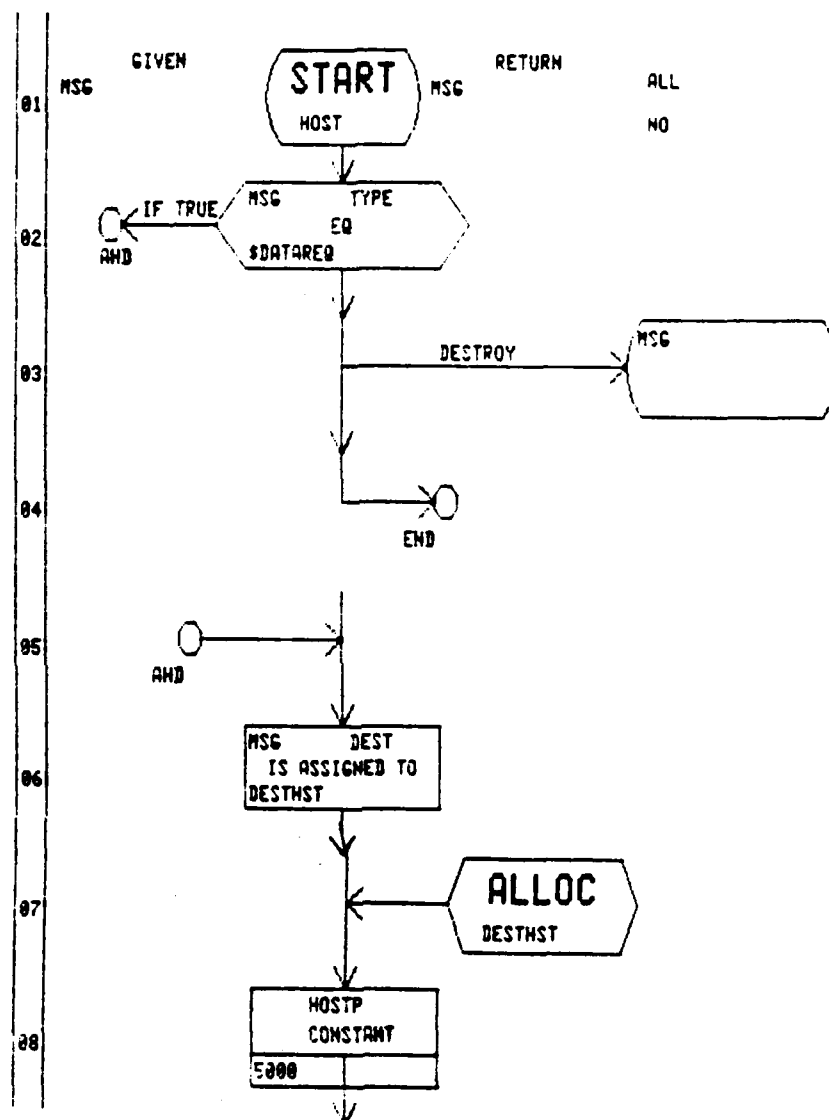


Figure 60. Example 3 HOST Data Message Handling

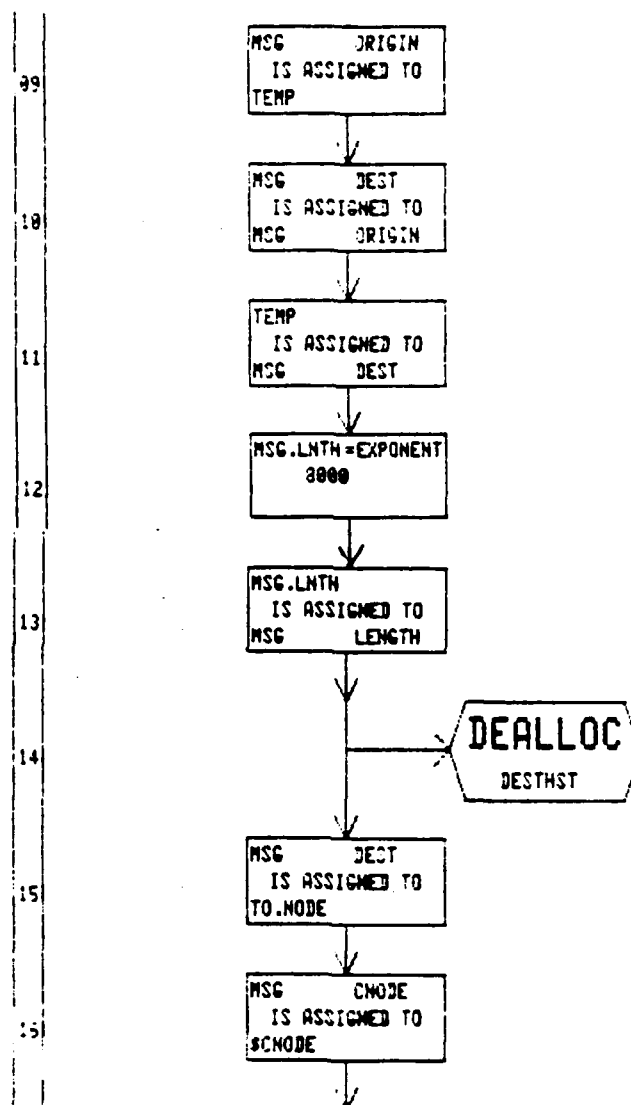


Figure 60. Example 3 HOST Data Message Handling (cont'd)

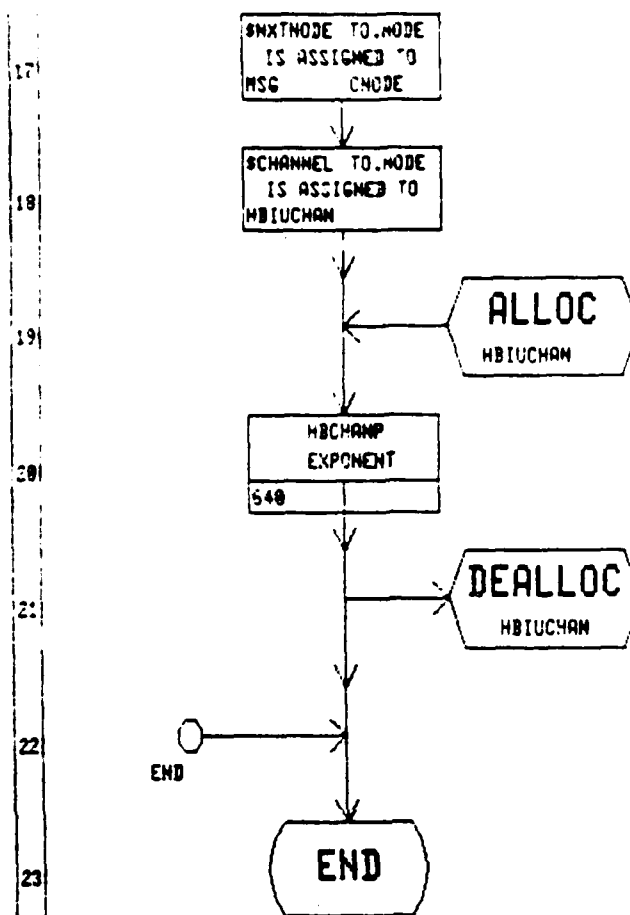


Figure 60. Example 3 HOST Data Message Handling (cont'd)

5.3.1.10 Process: DRHOST All activity in the bus model initiates with a data request at a host. The Process which performs the logic for this operation is called DRHOST. The Process DRHOST creates a data request message and is initiated with two parameters, ORIGIN which is the source of the data request, and DEST which is the destination node. The data request message is transmitted across the channel between the host and the bus interface unit and given to the Process BIU to be handled. The buffer resource name of the originating node host is embedded in message so that the network control element can allocate the buffers to receive data when the requested data message is received.

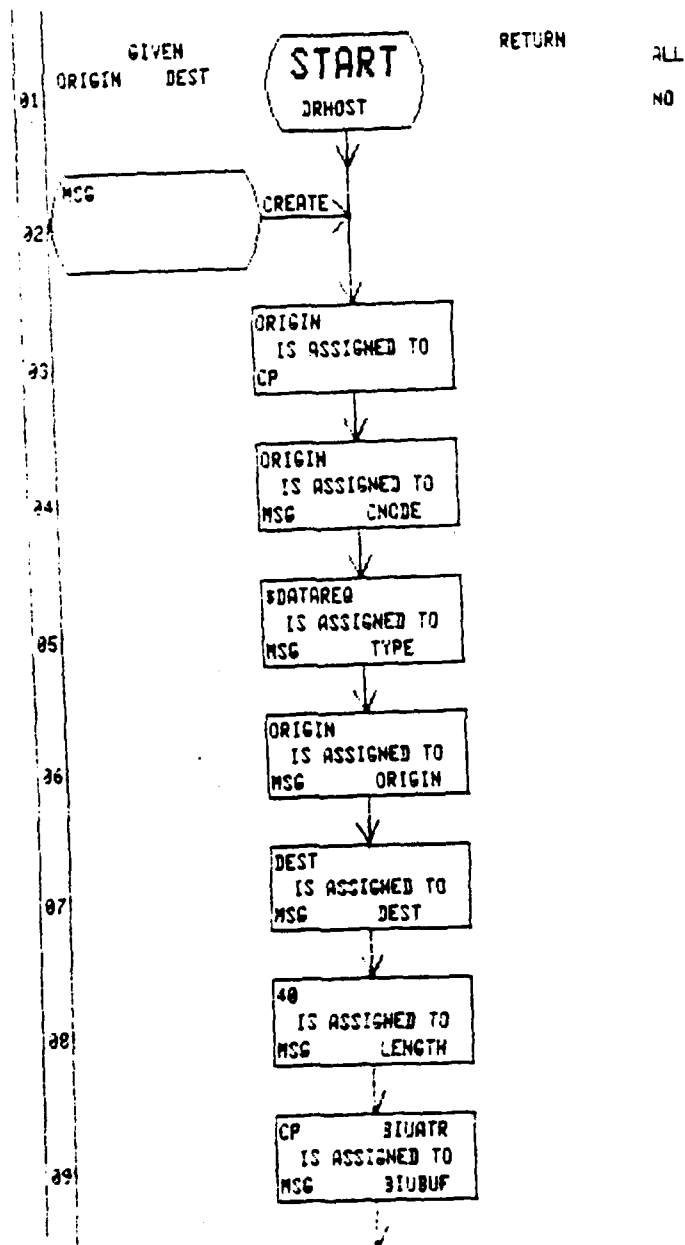


Figure 61. Example 3 Data Request Message Generation

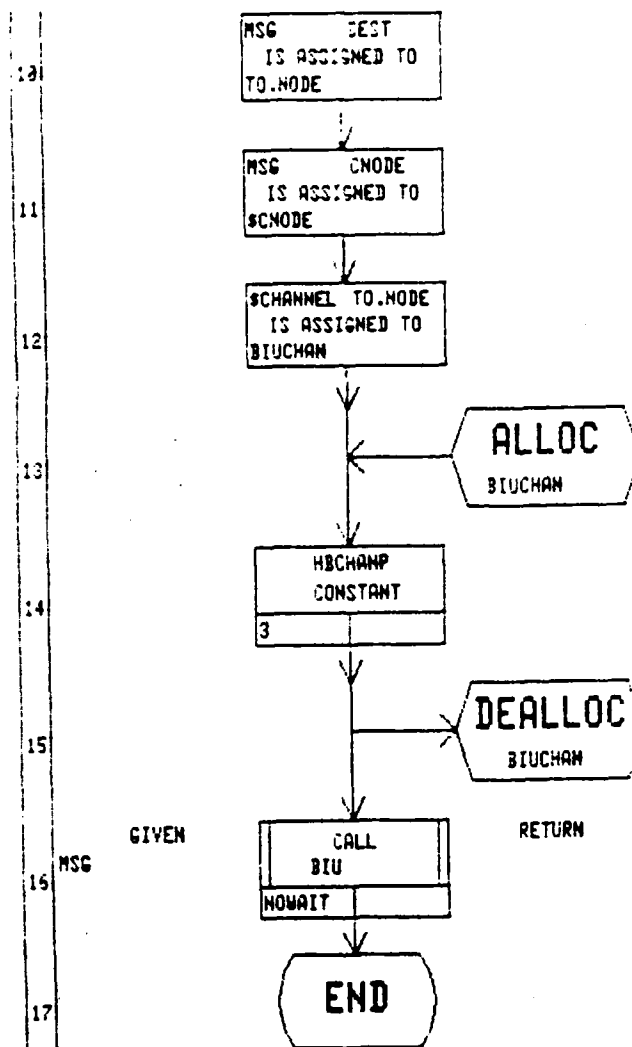


Figure 61. Example 3 Data Request Message Generation (cont'd)

5.3.2 Load Drivers: Processes FOHOST1 through FOHOST6 Simple Processes are used in this model to initiate data requests. These Processes are initiated by the Load, which assigns a current node attribute to the Process. The Process then initiates a data request from the current node to a specific host by calling the Process DRHOST providing a source node and a destination node. All Load Processes, FOHOST1, FOHOST2, FOHOST3, FOHOST4, FOHOST5, and FOHOST6, execute in nodes H1, H2, H3, H4,

45, and 46. This provides a mechanism for implementing a Load on the system described by a matrix of host nodes. The Process diagram for Process TCHOST1 is shown in an accompanying figure. All other Processes are similar with the exception that the destination node parameter in the call to DRHOST is different.

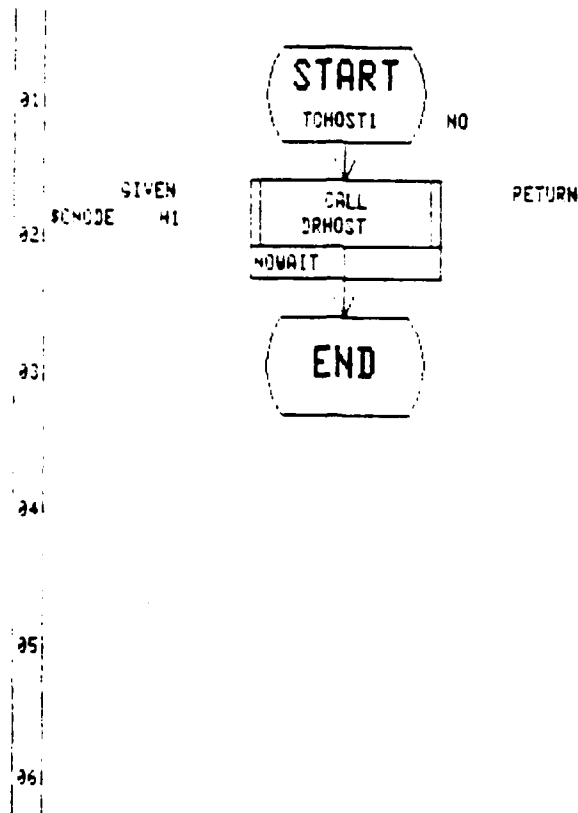


Figure 62. Example 3 Load Process TCHOST1

5.4 Analysis of Results

5.4.1 Performance Measure The model as build does not explicitly produce all the performance measures stated in section 5.1.2, but produces many statistics from which to compute those measures. Each host is represented by a Resource so it is possible to determine the utilization and queueing for each host under the Resource Report. Response time is tracked by the Item MSG, so the TIME IN SYSTEM statistics in the Item Report refer to this. The data bus and control bus are represented on the system architecture so they too are found in the Resource Report along with the simulation generated statistics.

5.4.2 Validation The easiest method for validating the bus model is by visual analysis of the Processes compared to the step by step description of the bus protocol given in Section 5.1.3. This indicates that the sequence of events represented in the

model matches the sequence of events indicated in the description of the protocol.

END

FILMED

3-84

DTIC